NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF NONDETERMINISTIC SEARCH PATTERNS FOR MINIMIZATION OF UAV COUNTER-TARGETING

by

Timothy S. Stevens

March 2013

Thesis Advisor: Timothy H. Chung
Second Reader: Michael Atkinson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
03-29-2013	Master's Thesis	2102-06-01—2104-10-31

4. TITLE AND SUBTITLE

Analysis of Nondeterministic Search Patterns for Minimization of UAV Counter-Targeting

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Timothy S. Stevens

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

In an attempt to mitigate the expanding counter-UAV capabilities of adversary countries developed in response to the United States' increased reliance on these platforms, we apply a nondeterministic search pattern to a finite area searcher. By implementing a Levy distribution on search leg lengths we analyze the trade-offs between efficiency and evasiveness of the searcher, comparing the expected time to target detection for a given set of Levy parameters to a probabilistic time to counter-targeting based on intelligence driven enemy capability. The culmination of this thesis is the development of a robust simulation tool, capable of modeling various parameters on both searcher and search area, the output of which is a quantifiable estimate on the probability of mission success.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	119	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ANALYSIS OF NONDETERMINISTIC SEARCH PATTERNS FOR MINIMIZATION OF UAV COUNTER-TARGETING

Timothy S. Stevens
Lieutenant, United States Navy
B.S., Computer Engineering, University of Arizona, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2013

Author: Timothy S. Stevens

Approved by: Dr. Timothy H. Chung
Thesis Advisor

Dr. Michael Atkinson
Second Reader

Dr. Robert Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In an attempt to mitigate the expanding counter-UAV capabilities of adversary countries developed in response to the United States' increased reliance on these platforms, we apply a nondeterministic search pattern to a finite area searcher. By implementing a Levy distribution on search leg lengths we analyze the trade-offs between efficiency and evasiveness of the searcher, comparing the expected time to target detection for a given set of Levy parameters to a probabilistic time to counter-targeting based on intelligence driven enemy capability. The culmination of this thesis is the development of a robust simulation tool, capable of modeling various parameters on both searcher and search area, the output of which is a quantifiable estimate on the probability of mission success.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

A2AD Anti-access/Area Denial
AOU Area of Uncertainty
ARSENL Advanced Robotic Systems Engineering Laboratory
CDF Cumulative Distribution Function
CRUSER Consortium of Robotics and Unmanned Systems Education and Research
DHS Department of Homeland Security
FOC Farthest On Circle
JIFX Joint Inter-agency Field Exploration
MANPADS Man-portable Air Defense System
NPS Naval Postgraduate School
PDF Probability Distribution Function
SLS Sea Level Standard
UAV Unmanned Aerial Vehicle
UAS Unmanned Aerial System
USG United States Government

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Unmanned Aerial Vehicles (UAVs) have become a mainstay of modern day military operations, such as surveillance and reconnaissance missions. Adversaries of the United States are fully aware of this shift and are developing weapons and training to counter these unmanned assets. Many of the counter-UAV weapons under development and testing require that a target lock be maintained on the UAV for some minimum amount of time. By randomizing the flight pattern, e.g., when conducting ISR missions, and limiting the time the UAV travels on any single flight leg, we can minimize the vulnerability of these assets. To accomplish this randomization we employ a Levy distribution function to determine the length of each search leg, while changes in searcher heading are drawn from a uniform distribution. We model realistic flight limitations using Dubins curves, which define the minimum distance path between two points of different heading orientation given the minimum turn radius capability of the searcher. Regression analysis of simulated search times is used to derive the expected coverage rate. We define a metric, the probability of mission success, comprised of a time to target detection by the searcher and a time to counter-targeting of the searcher by the adversary. A Bayesian update scheme is applied to the search to incorporate imperfections in sensor performance, along with a looping search function. Should no target detection occur within a specified amount of time, the searcher's travel will be biased toward the area of highest target probability density. The culmination of this thesis is the development of a simulation model for analysis on the employment of nondeterministic search patterns as a means to mitigate counter-targeting and counterdetection threats.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

I would like to first thank my wife, Kristy, and my children, Aiden and Leia, for their patience and endurance through not only this thesis process, but my entire naval career. I would not be where or who I am today without them.

I would, of course, like to thank my advisor, Tim Chung, for his zen-like patience and professionalism. This was a challenging and rewarding process, and I am grateful for the steady-handed guidance you provided throughout. I am extremely proud of the final product we achieved.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Background

Unmanned aerial vehicles (UAVs) have become the mainstay of modern-day intelligence gathering for military and government agencies worldwide. These assets provide valuable real-time situational awareness while minimizing several of the risks associated with personnel in potentially adversarial territories. A variety of mathematical models for searching such areas have historically been applied to provide decision support for these information-gathering missions. However, many search models and associated measures of performance address only the effectiveness of the search and deal solely with the constraints of the searcher's characteristics, such as limited endurance or limited sensor performance.

Instead, the *mission* performance may require that the employed search model address additional concerns that are relevant to realistic mission scenarios, such as the need to operate in unfriendly or clandestine environments and avoid counter-detection and/or counter-targeting. The ability to encompass mission-relevant metrics in a holistic mathematical model, rather than to construct individual models for different components of a complex mission, offers significant benefits to decision makers who must assess the relevant trade-offs.

For example, although maximal area coverage is essential, as nations develop more advanced air tracking and defense systems, the need for stealth by either platform or employment of randomized flight patterns that incorporate unpredictability (to avoid detection and targeting by the adversary) also increases, requiring a balance between search effectiveness and searcher evasiveness. Threats from emerging technologies such as directed energy weapons [1, 2], as well as conventional ones such as shoulder-launched rockets, pose potential danger to aerial assets conducting surveillance or other stand-off missions.

In this study, we analyze the employment of a nondeterministic search pattern over a finite area using randomly distributed leg lengths and heading angles. We are interested in the probability of mission success, i.e., the ability of the searcher to discover a target located in the finite area prior to being counter-targeted by the adversary.

1.2 Research Objective

The focus of this thesis is the development of an analytic model and associated computer simulation with which to analyze the area coverage and target detection capabilities of an unmanned searcher conducting randomized search over a finite area. The effectiveness of such a search is measured against the known efficiency of a continuous sweeping search by deriving equations representing the probability of detecting a target uniformly distributed within the finite area, as well as the expected time elapsed prior to the detection. Utilizing this estimated time to target detection we apply assumptions regarding adversary targeting capabilities to formulate a probability of mission success, e.g., the probability that the time required for the searcher to detect the target is less than the time required for an enemy to counter-target the searcher.

1.3 Literature Review

The field of robotic coverage and search has an extensive history, including substantial efforts to optimize the coverage patterns for different objective functions such as minimum time or coverage path, minimal overlap, increased robustness to localization errors, etc. [3–5]. The role of randomized coverage and search patterns has also been explored, with studies ranging from optimal random paths to traverse all nodes of a finite network, to random walks over infinitely two-dimensional area [6–10]. Specifically, several studies have been conducted over the years in an effort to determine the proper distribution of search leg lengths resulting in an efficient random search, where efficiency is measured by the searcher's ability to cover a large area in a finite period of time [11,12]. The demonstration of this type of efficient random search in nature has been the focus of multi-disciplinary efforts with animal behavior modeling, e.g., [13–17].

Viswanathan et al. recognized this efficiency in the natural foraging habits of animals, determining that several species exhibit Levy characteristics in their search patterns [18, 19]. Given that the livelihood of these species depend on efficient search techniques for survival, the mimicking of these habits should translate to an efficient (if not optimal) search pattern for most object search scenarios. Viswanathan et al. analyzed the search leg distances recorded during a three-month observation of albatross (five albatross conducting nineteen foraging journeys) [20]. Unlike what might be observed through demonstration of a random Gaussian or Raleigh distribution, no well-defined variance was observed among the search leg distances. Plots of the recorded data indicated a long-tailed power-law distribution of lengths corresponding to Levy flight motions [18]. In particular, the observed distribution of flight lengths was

characterized by

$$P(t_i) \sim t_i^{-\mu}, \tag{1.1}$$

where t_i is the time between landings, and μ represents the fractal dimension of the long-tailed power-distribution with $1 < \mu \leq 3$. The distribution converges to a Gaussian for $\mu > 3$ and does not correspond to a normalizable density function for $\mu \leq 1$. Histogram plots of the nineteen individual foraging trips suggested an average $\mu \sim 2$.

Further works by Viswanathan et al. assert that there are *several* animal foraging patterns observed in nature (albatross, honeybees, aphids, drosophila, etc.) which mimic a Levy flight with $\mu \sim 2$. The authors credit these multiple observances of the Levy distribution in nature to the fractal properties of the Levy search that lead to a large dispersion from the origin over a finite time period. This characteristic is particularly beneficial for swarms or groups of N Levy walkers versus N Brownian walkers [21]. The Levy walkers are able to cover a more widely spread area and diffuse more quickly, thus limiting competition between individual searchers. Conversely, Brownian searchers tend to remain clustered together for longer periods of time. In summary, it is shown that due to the low probability of revisiting a previously searched area, the Levy flight possesses an advantage over Brownian motion when the target density is sparse and the targets are randomly distributed, but makes little difference in small areas of high target density [18].

The authors' extended analysis of the Levy search differentiates between destructive and non-destructive search methods, the former being one in which an area is precluded from future searches once it has been visited, the latter allowing for multiple visits to any given site. A set of simulations is conducted in which the searcher is attributed a fixed search radius representative of an animal's visual range, conducting a continuous search while moving along the Levy path [19]. If a target is detected within this visual range, the searcher can change direction and move directly to the target, eliminating the target from the search area. If no target is detected within the visual range of the searcher, or if successful target elimination is achieved, a random heading from a uniform distribution and a random search length from the power-tail Levy distribution are drawn and the searcher moves along this new path until a target is acquired or the end of that search leg is reached. Through comparison of the simulation results to the foraging data collected on honeybees, deer, and wandering albatross, the authors affirm the observance of $\mu \sim 2$ for any search dimension where there exists no *a priori* knowledge of target location [18, 19].

Raposo et al. expand upon the work of Viswanathan et al. by adding a parameter, τ, representing a regeneration time during which a visited site is not available for revisit. The authors show that for any power-law tailed distribution (Equation 1.1), with the optimal exponent restricted to $1 < \mu_{opt} \leq 2$, by relating the velocity of the forager, v, and the recovery time of the site, τ, such that $z = \frac{v\tau}{\lambda}$, where λ is the average distance (based on a spatial Poisson distribution of target location) between targets, the optimal search strategy is defined by $\mu_{opt} = 2 + \frac{2}{\ln z} + O\left(\frac{1}{\ln z}\right)$. As τ approaches 0 (non-destructive limit), the best search strategy is observed with $\mu_{opt} \sim 2$. This coincides with the conclusions of Viswanathan et al. As τ approaches infinity (destructive limit), the best search strategy is observed using $\mu_{opt} \to 1$, suggesting upper and lower bounds on the mean flight length for a Levy search pattern [22].

Calitoiu and Milici demonstrate that the Levy distribution exhibits characteristics advantageous to groups or swarms of animals foraging over large areas [23]. The authors describe the benefits of the Levy flight over the Brownian walk as being derived from the fact that the search leg distances associated with Brownian motion are scaled to a set magnitude defined by the searcher, where as the Levy flight lengths are a function of target spacing. This is observed in nature through a species' ability to adjust to changing environmental conditions and availability of food/prey. The authors employ the Levy flight pattern to simulate destructive and non-destructive searches using non-cooperative agents; searchers working independently to locate targets in a common area with the constraint that two agents cannot occupy the same vicinity at the same time. The authors draw a specific distinction between the utilization of a Levy flight vice a Levy walk for the simulation. The term "Levy flight" is used to describe a pattern in which the search is conducted only at the ends of each leg; searchers use a "move, stop and search" technique. In the Levy walk, the searchers are continually able to detect a target using a "move and search, stop" technique. For both the destructive and non-destructive simulations, a shift parameter of $\mu = 2$ is used. The Levy distribution searches are compared to searches utilizing a step size equal to unity, consistent with Brownian motion. In both destructive and non-destructive simulations the Levy searchers traverse distances farther from the origin. However, because of the non-continuous search characteristics exhibited during the Levy flight, the Brownian searchers are able to search more total area. The results of these simulations infer that a Levy distribution of search leg lengths results in a dispersion from the origin that is beneficial in areas of low target density. The employment of a Levy walk vice Levy flight would further increase the effectiveness of this search.

Each of the aforementioned works show that a Levy distribution of search leg lengths often

results in a more rapid dispersion from the origin when compared to Gaussian-based Brownian motion, leading to a more expanded area of coverage and less local competition between entities searching as a group. It is this dynamic aspect of the Levy distribution that makes it an intriguing candidate for use in randomization of a search pattern over a finite area.

While widely studied with regard to natural behaviors of animal species, the Levy distribution is also the focus of multiple robotics-based search strategies, being applied to search leg lengths of autonomous searchers attempting to located targets in a minimal duration of time [24, 25]. Lenagh and Dasgupta conduct simulations utilizing what they describe as linear Levy and looped Levy search patterns (described shortly) employing multiple searchers to detect a single target located within an unbound search area [26]. The linear Levy search employs search leg distances drawn from the closed-form Levy distribution of Equation 1.2 and illustrated in Figure 1.1. The searcher travels these Levy lengths conducting a continuous search for the target, consistent with the Levy walk described previously.

$$L(l_j;\mu,c) = \sqrt{\frac{c}{2\pi}}\frac{e^{\frac{-c}{2(l_j-\mu)}}}{(l_j-\mu)^{\frac{3}{2}}} \tag{1.2}$$

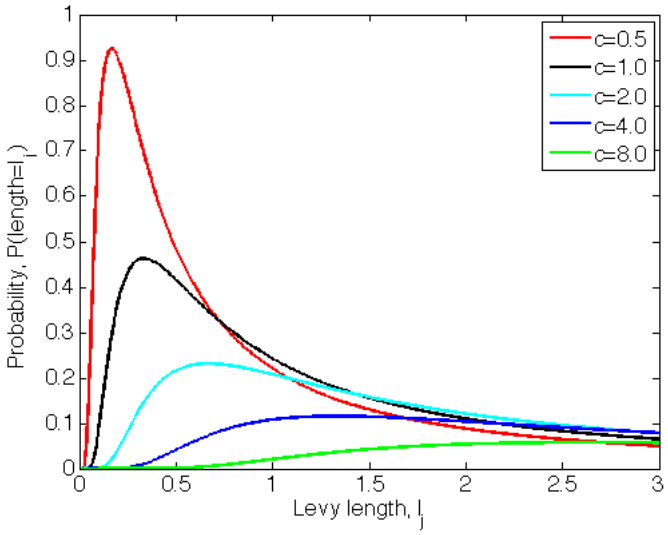


Figure 1.1: Probability distribution for the closed-form Levy with shift parameter $\mu = 0$. The smaller the scaling parameter, c, the sharper the probability peak and greater chance of returning shorter lengths.

In this closed-form Levy distribution $L(l_j;\mu,c)$, the variable μ represents the shift or location parameter for the distribution of lengths for leg l_j, defining the minimum length returned. The c

represents the scaling parameter controlling the height of the curve, and thus the probability of any particular length being returned. Figure 1.1 shows that for smaller values of c the probability peak of the distribution is higher, such that the probability of obtaining a value close to the parameter μ is greater [26]. The authors define the change in orientation of the searcher only to occur at angles greater than $\frac{\pi}{2}$ from the current heading, thus the initial heading of the searcher is random, drawn from the uniform distribution, $U(0,2\pi)$, with subsequent turn directions defined as $\frac{\pi}{2}+\theta$, where $\theta \sim U(0,\pi)$. The looped Levy search utilizes the same distributions; however, if contact is not gained/held on any target after a finite time period, the searcher will return to its origin (if no target information is retained) or to the area of last known target location (if contact was held but subsequently lost) and recommence the search in a new random direction. The searcher is equipped with a perfect sensor, such that if the target is within a distance less than the sensor range from the searcher, the target is located. These simulations are conducted in three phases, the first being search for a stationary target that has relocated to a spot some distance from its last known location. The remaining phases involve localizing (phase two) and tracking (phase three) of mobile targets. Each of these simulations is conducted for varying values of parameters μ and c. The authors conclude that little difference exists between the performance of the looped and linear Levy models while searching for stationary targets. The functionality of the looped Levy to return to the last known target position limits the time the searcher travels a path of zero target density, resulting in increased performance over the linear Levy when searching for moving targets. Throughout the tracking phase the looped Levy search demonstrates a timelier re-acquisition of a moving target after contact is lost, and for all simulations, faster target detection occurs when the value of the shift parameter μ is set to equal the initial target distance from the searcher (though this requires *a priori* knowledge of the target's location) [26]. This distribution function representing the Levy walk is the foundation of the non-deterministic search developed in this thesis. The target locations are randomly selected and unknown by the searcher, thus dictating the need for further analysis regarding the optimal shift (μ) and scaling (c) parameter values.

Mantegna and Stanley explore the significance of Levy movements within bounded physical systems, such as turbulent fluids or single molecules embedded in a solid, in which there exists an unavoidable cutoff in distance traveled [27]. This research provides insight as the the effects of conducting Levy-based searches within a bounded area of operations. The authors refer to the random movement within systems of this type as truncated Levy flights and derive a mean and variance for this truncated Levy distribution for which the mover is physically limited in

distances it can move. Mantegna and Stanley conclude that, unlike the standard Levy, the distribution of this truncated Levy does converge to a Gaussian distribution (i.e., exhibits behavior prescribed by the Central Limit Theorem) when the sample size exceeds 10^4 to 10^5.

As referenced, there have been several studies addressing the optimization of random search patterns, i.e., a series of random straight line distances linked by random changes in direction. The majority of these studies establish evidence that the rapid dispersion characteristic of the Levy distribution provides the advantageous opportunity to achieve maximum area coverage with minimal revisit time. These previous works provide the initial parameters for our model and simulation. This thesis builds upon these studies by utilizing the closed-form Levy distribution (Equation 1.2) to simulate and analyze the coverage ratio, that is, the area searched as a function of time of flight, and probability of detection for a Levy walk pattern over a bounded area. To model the realism of UAV movement and sensor capabilities, several other works are reviewed.

In 1957, Lester Dubins proved that the shortest distance between two points in two-dimensional space, each defined by a coordinate and heading, can be expressed as a single path configuration of the set $\mathbb{D} = \{CSC, CCC\}$, where C represents a curve of minimum turn radius and S represents a straight line segment [28]. These can be specified further such that the shortest path between the two points must consist of some path configuration D, where $D \in \mathbb{D} = \{RSL, RSR, LSL, LSR, RLR, LRL\}$, with R and L representing right and left turns, respectively. By calculating the length of each of the three path segments, the optimal path, i.e., that with the minimum overall distance, can be determined. This path is known today as "Dubins path." This work is extended in the development and analysis of Reeds car, a Dubins vehicle allowing for both forward and backward motion, and the Dubins airplane, adding altitude as a functional parameter in the determination of the Dubins path [29–32]. Parlangeli et al. further the work on the Dubins model by introducing a path comprising various waypoints, each of which must be visited, in order, by a mover bound by limited curvature [33].

Shkel and Lumelsky simplify the mathematics behind Dubins curves by proposing a method which allows for the selection of the shortest Dubins path from the set $\mathbb{D}$ without having to conduct the cumbersome segment length calculations performed by Dubins [34]. Using four quadrants akin to that of a polar coordinate system, the authors distinguish *classes* of initial and final heading configurations, realizing that not every path in $\mathbb{D}$ is feasible for every configuration of headings. Further, these *classes* are divided into *equivalency groups*, which account for

equal but opposite flight paths of orthogonal pairs of heading configurations. For example, a searcher at an initial point with heading $\frac{\pi}{2}$ moving to a final point with heading 0 would conduct a *RSL* flight path. Had the initial heading been $-\frac{\pi}{2}$ the searcher would travel the same distance but opposite path, or *LSR*. The authors define sixteen *classes* arranged in a four-by-four matrix of quadrants, with each row representing the quadrant of the initial heading and each column representing the quadrant of the final heading. The intersection of the rows and columns contain information regarding the appropriate Dubins path, eliminating several of the previously required calculations. All calculations conducted by Shkel and Lumesky assume a unit turn radius in order to simplify the results; thus the scalability of their equations must be analyzed in order to be applied to this thesis, in which a non-unit or variable turn radius can be set by the user of the simulation.

The operational characteristics of the sensor utilized by the searcher necessarily affects the performance of the search strategy regardless of pattern employed. It is essential for operational considerations to account for possible imperfections in sensor performance in order to obtain a full appreciation for any search pattern efficiency. Several works address the significance of these sensor imperfections, i.e., false and missed detections [35, 36]. Kress et al. established a discretized search grid with each cell assigned a prior probability of containing a target and no cell capable of housing more than one target [37]. They simulate a search utilizing probability thresholds in order to determine if a cell is clear or contains a target. A Bayesian update process, incorporating the probability of false alarms and missed detections, is used to determine posterior probabilities of target location. The simulated UAV searchers used in this previous work are non-adaptive, meaning that they can not adjust the predetermined flight path to accommodate for the updated probabilities. To account for this, the searchers are allotted a specific number of looks, each of which is used only if probability of a target being located in the visited cell does not exceed the threshold for positive detection or clearance of the cell. Chung and Burdick also incorporate a Bayesian update scheme into a discretized search pattern, applying a "decision-making" process to searcher movement. The discrete Bayesian update process is combined with a look-ahead search algorithm to bias the searcher movement toward a region within the discretized area presenting the highest belief of target presence [38]. The incorporation of this "decision making" process and Bayesian update of the search grid are utilized in this thesis to implement a looped Levy search, directing the searcher to an area of highest target probability if no target is found within a finite time period.

1.4 Scope, Limitations, and Assumptions

The purpose of this thesis is the mathematical derivation and subsequent computer simulation of a nondeterministic search algorithm that maximizes the area covered during the searcher's flight while limiting the probability of counter targeting. The scope of this thesis is limited to answering the following questions. Can we:

- determine an optimal distribution of random flight leg distances and turn angles which improves upon the coverage ratio achieved during a continuous random search, while at the same time reducing the single leg counter-targeting exposure of the searcher?
- develop a tangible and effective means of measuring counter-targeting avoidance?
- develop a scalable nondeterministic search algorithm capable of implementation via any unmanned system (whether aerial [UAV], ground [UGV], surface [USV] or subsurface [UUV])?
- derive a single specific adjustment parameter that can scale the model for any of the above search platforms?
- define and achieve an acceptable coverage ratio and probability of detection observable over multiple simulated scenarios, and then use these observed characteristics of the model to derive a probability of achieving a given coverage ratio or probability of detection for various scenarios?

For the purposes of this research, it is assumed that the searchers are "on station" at the onset of the search pattern (i.e., take-off and landing of the UAVs are not accounted for). The model addresses the realistic flight limitations of a UAV by limiting the searcher to travel along Dubins paths, with all heading changes limited to a minimum turn radius. Environmental effects such as wind/current are assumed to have no effect on the flight trajectory of the searcher. Other than directional changes, the motion characteristics of the searcher will remain fixed (i.e., no speed or altitude changes) throughout the entirety of the search pattern. It is further assumed that the search sensors are mounted on some form of gimbal device, such that changes in vehicle list and trim that would otherwise shift the sensor footprint need not be modeled. With respect to the search area, it is assumed that no obstacles or terrain obstructions exist that would interfere with the path nor sensor of the searcher. When simulating multiple searchers, each unit is able to maintain constant and perfect communications with all other units and the central controlling station. Finally, it is assumed that at most one target exists per each discretized cell of the search area.

1.5 Contributions of the Thesis

The research accomplished during the development of this thesis is beneficial to the United States military and other government agencies operating unmanned vehicles in support of intelligence, surveillance, and reconnaissance missions. The main contribution of this paper is the development of a probabilistic model by which to analyze the trade offs present when using an autonomous searcher to find a stationary target in minimal time while avoiding counter-targeting by the adversary. We define a novel performance measure which captures this balance between search effectiveness and evasiveness of the searcher, which can be used as a basis for assessing search mission performance in bounded adversarial environments. We utilize a Levy distribution in the generation of nondeterministic search paths, and highlight the insights arising from the analysis of the search and counter-targeting performances.

Further, this thesis research represents the author's selection as a recipient of the Space and Naval Warfare Systems Center, Pacific (SSC Pacific) NPS Student Fellowship Program, which enables interactions with SSC Pacific laboratory researchers, and also provides substantial equipment and travel funds in support of enhancing the student's research.

Additionally, a portion of the thesis work presented herein, describing the use of the Levy-distributed search leg lengths with an instantaneously turning searcher, has been accepted via peer-review for publication and presentation at the 2013 IEEE International Conference on Robotics and Automation (ICRA) in Karlsruhe, Germany [39]. This conference represents a flagship venue for dissemination of advanced research in the international robotics and automation communities.

1.6 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we include a detailed definition of all parameters and assumptions utilized in the model and simulation. The derivation of the analytic model to determine coverage ratio with respect to time over a finite search area is addressed. Chapter 3 details the development and implementation of the computer simulation based on the analytic model, to include the incorporation of Dubins paths, Bayesian probability updates, and the looping functionality of the search. Chapter 4 analyzes the results of the simulation in comparison to expected values calculated in Chapter 2. Sensitivity analysis is performed on each of the adjustable input parameters of the model, in particular the shift and scaling parameters of the Levy distribution and the size of the search area and sensor footprint.

In Chapter 5, the final conclusions are revealed, to include recommendations on the optimized values of the model parameters required to achieve the desired balance of searcher coverage and stealth. In this chapter we also address recommendations for follow-on studies with regard to the Levy search pattern.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Mathematical Formulation

This chapter describes the mathematical mechanisms and theory behind the Levy-based search pattern that is the cornerstone of this research. The reasons for the establishment of a finite search space to include the operational significance of the shape and size of the area are addressed. A detailed description of the searcher movement and sensor characteristics based on parameters of existing real-world unmanned aerial craft is included. This chapter also describes the implementation of a Bayesian updating scheme in which *prior* and *posterior* probabilities of target presence are calculated based on sensor operating characteristics, contributing to the efficiency of target detection.

2.1 Background

This research can best be summarized through description of a search scenario encountered often during military operations. Assume an intelligence report is received by an operational unit indicating that a high value target is, or recently was observed, conducting operations at the location indicated in Figure 2.1. It is desired that the unit conduct a UAV search to confirm presence of the target for facilitation of a possible strike mission. The unit is not to violate the airspace of Country X, as indicated by the shaded area in the upper left. To further complicate the situation, Country X has established several anti-UAV assets along its borders and in the vicinity of the last known target position. Successful target detection requires that the unmanned searcher execute an efficient search while maintaining enough randomness of flight to avoid being counter-targeted and destroyed. A common measure of efficiency in search is the coverage rate of the searcher, or how much of the area can be searched in a given amount of time. In general, this rate, $\dot{F}(t)$, is a function of the speed and sensor footprint of the searcher,

$$\dot{F}(t) \triangleq \frac{dF(t)}{dt} = vw, \tag{2.1}$$

where v is the searcher velocity and w is the search sweep width of the sensor [40]. The most efficient search utilizes a sweeping pattern of area coverage as shown in Figure 2.2. Assuming the searcher is equipped with a perfect sensor, meaning that any target located within the maximum range, r_{sensor}, of the sensor is detected, the sweep patterns are such that no gap (missed coverage) and no overlap (excess coverage) exists between sweeps. For this searcher with a

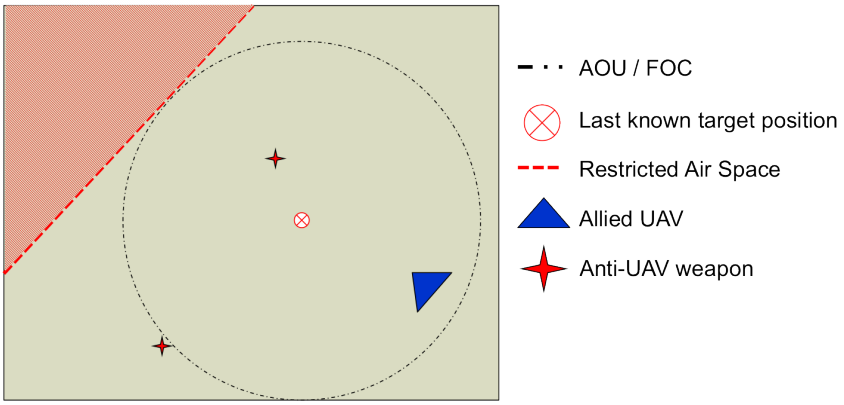
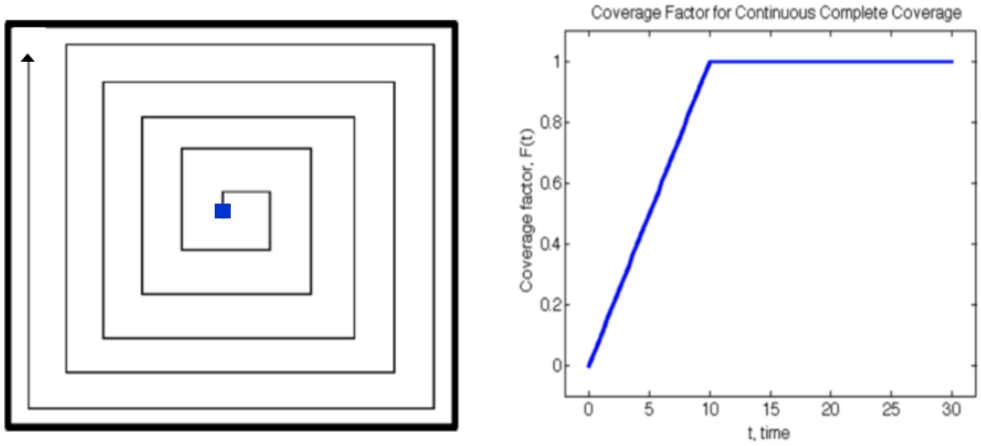


Figure 2.1: The purpose of this research is to develop a nondeterministic search pattern capable of efficiently locating a target while minimizing vulnerability to enemy attack.

perfect sensor conducting a continuous non-overlapping search of the total area, the coverage ratio with respect to time, $F(t)$, is defined as

$$F(t) = \frac{vwt}{A_{search}} \tag{2.2}$$

where t is time and A_{search} is the total area to be searched. Thus, at any time t, $F(t)$ represents the percentage of A_{search} searched. This continuous ideal search is an important benchmark in that no segment of the search area is missed and no segment is searched more than once, resulting in the efficient linear rise to total coverage shown in Figure 2.2.



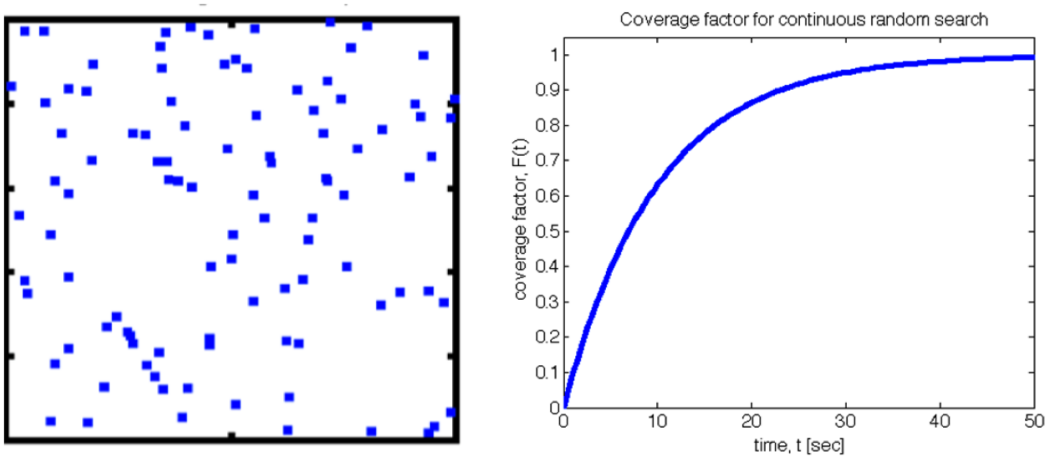
(a) (b)

Figure 2.2: (a) An example of perfect non-overlapping complete search, commonly referred to as the "lawnmower" or "spiral-out" pattern. (b) The resultant linear increase to total area coverage exhibited by the "spiral-out" search pattern. From [41].

In contrast to the continuous ideal search is the continuous random search pattern. Random movement over the search area coupled with discrete "looks" results in an inefficient search pattern albeit asymptotically complete. This search is often likened to dropping equally sized pieces of confetti onto a surface, one at a time, until the entire surface is covered [40]. As with the ideal search, the time to complete coverage is a time-based function of the searcher speed and sweep width. However, because the movement over the search area is random, the potential exists for revisit of any segment in the area, resulting in a logarithmic shape to the coverage ratio over time. As more of the area is searched, the probability of a random movement to an area not previously covered decreases and the coverage exhibits a slow and asymptotic rise with "diminishing returns" to complete coverage. The equation for this continuous random coverage ratio is defined as:

$$F(t) = 1 - e^{-\frac{vwt}{A_{search}}}, \tag{2.3}$$

with *vwt* determining the size of the "confetti piece" uncovered with each "look" [40]. Illustrations of this search pattern and the resulting exponential shape of the coverage ratio due to the potential for revisit are shown in Figure 2.3.



(a) (b)

Figure 2.3: (a) An example of the continuous random search, commonly referred to as the "confetti search." (b) The resultant exponential increase to asymptotically complete coverage of the total area exhibited by the "confetti search" pattern. From [42].

The continuous ideal search pattern demonstrates great *exploration* potential, efficiently covering the search area in a minimal amount of time. However, the employment of a perfect sensor required to achieve coverage which exhibits no overlap and no excess is unrealistic when translating to real-world applications. The continuous random search shows great *evasiveness*

capability due to the unpredictable nature of the random movement, but the requirement to instantaneously move about the search area is again not transferable to real-world applications. The intent of this research is to design a search algorithm that promotes a balance between the efficient exploration characteristics of the ideal search and the evasion potential of the searcher as exhibited by the random pattern. This hybrid pattern can then be executed to perform the mission objective: maximize the probability of the UAV searcher detecting the target prior to being counter-targeted and destroyed by the anti-UAV assets (Figure 2.4).

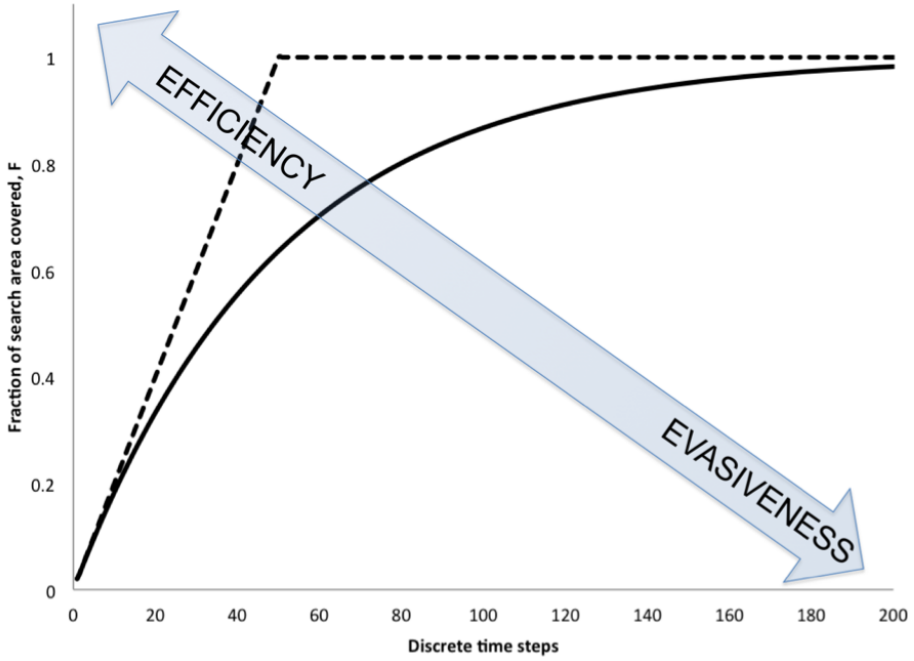


Figure 2.4: The goal of this research is to define a realistic and realizable search pattern promoting the *efficacy* of the ideal search (dotted) while minimizing the *evasiveness* of the searcher through incorporation of random movement.

We develop a model by which to quantify this expected probability of mission success for the unmanned unit conducting a nondeterministic area search within some bounded search region. We apply tangible measures of performance (MOP) to this search mission in order to quantify the probability of successfully detecting a target. We define a time to target detection (T_D) as the time elapsed from the start of the search until positive detection of the target. This parameter is a function of the searcher's movement and sensor capabilities, the size of the search area, and the random distributions of heading angle and transit leg length used to govern the trajectory. An estimated time to counter-targeting (T_C) is derived from models of intelligence assessments of

adversarial technologies operating in and around the search area. This intelligence assessment consists of the estimated time required for an anti-UAV weapons system to achieve a target lock and subsequent firing solution on the searcher. T_C is the time elapsed from the start of the search until the time the searcher is destroyed due to traveling a predictable path, such as straight flight legs, for a time exceeding the assessed enemy capability. Given that T_C is a function of the counter-targeting capacity of the adversary, and operationally beyond our control, we treat it as a random variable, establishing distributions on counter-target times specific to various levels of capability. The total probability of mission success can then be derived from these two parameters, where mission performance is quantifiable, defined as the expected time to target detection being less than the time for counter-targeting of the searcher:

$$P(T_D < T_C) \qquad \text{Measure of Performance} \tag{2.4}$$

2.2 The Search Area and Searching Unit

2.2.1 Establishment of a Finite Search Area

This research utilizes a finite search area designed to represent a circular area of uncertainty (AOU). An AOU is used in contact management situations when uncertain or time-late intelligence regarding a target's position is obtained, or when contact on a known target is lost. The shape of the AOU is often circular or elliptical, with the boundaries defined by factors ranging from confidence in the intelligence source to the attributes of the target itself. Often the radius of this AOU is a function of the time late condition of the intelligence report and the maximum transit speed of the target, with the boundary referred to as the farthest-on circle (FOC). In other words, the size of the AOU is dependent on how far the target could have possibly traveled in the time since the location was reported. In many cases, it is probable that the target is not transiting at maximum capable speed on a single straight-line heading, and thus, the established AOU is a reasonable and realistic upper bound on the search area. The size and shape of the AOU may also be a product of the confidence in and fidelity of the intelligence report. Other factors such as how long, i.e., time late, it takes to position a searcher in the area must also be taken into account and can often increase the size of the AOU.

In many scenarios, the air space or ground territory where the search is conducted poses sensitive territorial issues, requiring the establishment of a strict search boundary. To model this operationally relevant constraint, we establish a boundary zone, shown in Figure 2.5. The width

of this boundary zone is equivalent to twice the turn radius ($2r_{turn}$) of the searcher. The maximum straight distance between the initial and final waypoints, (x_o, y_o) and (x_f, y_f), respectively, of any search leg, will not exceed this artificial boundary, ensuring that the searcher will never violate the actual area boundary. This constraint still enables the searcher to move over the entirety of the search area. The parameters associated with the search area are summarized in Table 2.1.

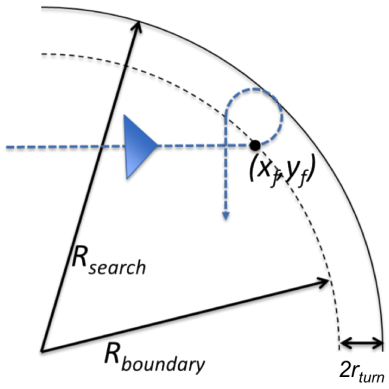


Figure 2.5: A boundary radius is established within the circular search area, ensuring that the searcher does not violate the actual area boundary.

Variable	Definition	Units
A_{search}	total search area	[meters^2]
R_{search}	radius of the circular search area	[meters]
$R_{boundary}$	effective radius of the search area, outside of which the UAV must be turning	[meters]

Table 2.1: Variables associated with sensor operations

2.2.2 The Searcher

Searcher Movement

The searcher modeled throughout this research is intended to represent UAV technology currently in service, in particular, mid-sized platforms utilized by Special Forces and other forward deployed personnel. These systems are often rugged, low-cost UAVs capable of short range search and small payload delivery missions. Table 2.2 summarizes common operating parameters exhibited by these mid-size platforms [43].

Parameter	Capability
Stall Speed	18 m/s (40 MPH)
Maximum Speed	36 m/s (80 MPH)
Cruise Speed	22 m/s (50 MPH)
Operational Maximum Range (still air)	320 km (200 miles)
Operational Endurance	4 hours

Table 2.2: General operating parameters of a mid-sized UAV, derived from [43, 44]

Sensor Footprint

It is assumed for the purposes of this research that the UAV is equipped with a sensor package capable of distinguishing between the target and the environment in which it is operating. We assume a circular sensor footprint for ease of calculations. The field of view (FOV), that is, the extent of the target area visible during any instant in time, is assumed to be 30 degrees. This is derived from the works and reports of the U.S. Army Research Center and the Defense Technical Information Center and considered to be the average capability of small and mid-sized UAVs [44]. The "on the ground" footprint of the sensor is determined using a height of eye formula (Equation 2.5) based on the searcher altitude (alt) to calculate the sensor radius, r_{sensor}, and thus, area of the circular footprint (Equation 2.6), as shown in Figure 2.6 and Table 2.3. This sensor range, r_{sensor}, when extended along both sides perpendicular to the searcher's trajectory, determines the sweep width, w, referred to in the search theory examples of section 2.1.

$$r_{sensor} = \frac{1}{2}(alt)\tan(FOV) \tag{2.5}$$

$$A_{sensor} = \pi r_{sensor}^2 \tag{2.6}$$

To simplify the promulgation of searcher and target position, as well as calculations on distance and probability of target presence, the search area is discretized into one meter by one meter square cells. Assuming no intelligence suggesting otherwise, each of these discretized cells has an equal opportunity of containing the target, assumed to be no larger than the one meter by one meter cell. As the UAV searcher moves over the search area, the sensor footprint uncovers these cells. For the purposes of this research, it is assumed that if the circular footprint of the searcher overlaps any part of a grid cell, the cell is considered searched. This assumption is plausible due to the large scale of the search area (generally of the magnitude 10^6 square meters) as compared to the area of the searcher footprint (on the order of 10^2 square meters). The possible over-accounting for the search of grid cells not completely covered by the sensor is therefore

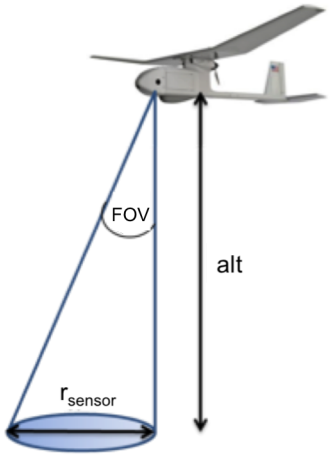


Figure 2.6: The search area uncovered during any discrete instant in time, A_{sensor}, is a function of the sensor radius, r_{sensor}, as determined by the sensor field of view, FOV, and the altitude of the searcher, alt.

negligible, yet greatly simplifies the calculations involved when computationally determining which cells have been searched.

Modeling of Sensor Imperfections

The assumption of a perfect sensor is sufficient when determining the effectiveness of the searcher movement pattern, but is not a feasible assumption for modeling real-world search dynamics. In assuming a perfect "cookie cutter" sensor, any target that lies within the maximum range of the sensor is seen. Due to adjustability in sensor gains and possible limitations in processing power for these mobile sensor systems, there exists, in reality, the tendency to generate false contact reports or missed target detections. We denote the probability that a sensor indicates the detection of a target when one is not present, generally referred to as a false alarm, by the designation α. The probability that the sensor fails to indicate the presence of a target when one is present, commonly referred to as a missed detection, is denoted β. A perfect sensor is characterized by both α and β equal to zero.

Throughout this research, it is assumed that the target exists within the confines of the circular search area; thus, the sum of the probabilities that the target exists in each grid cell must equal one, i.e.,

$$\sum_{i=1}^{n} p_i = 1, \tag{2.7}$$

where p_i is the probability that the target is present in cell i and n is the total number of grid cells in the search area. In the absence of any intelligence information indicating that the target

is in a certain part of the search area, the probability that the target is in each cell is initialized as

$$p_i = \frac{1}{n}, \quad \forall i = 1, \ldots, n, \tag{2.8}$$

to represent a uniform probability. To derive the probability that the target exists in each individual grid cell, these probabilities of false alarms and missed detections can be applied according to a Bayesian updating scheme. Bayesian updates use the *a priori* probability that the target exists in a cell, p_i, and apply the α and β sensor characteristics to determine an *a posteriori* probability the target exists. If a searcher detects a target in a given cell, the following update is applied to that cell's target probability [38]:

$$p\left(\begin{matrix}target\\present\end{matrix}\middle|\begin{matrix}\text{“}target\\detected\text{”}\end{matrix}\right) = \frac{(1-\beta)p_i}{(\alpha)(1-p_i)+(1-\beta)p_i}. \tag{2.9}$$

As more detections occur in that cell, p_i increases per Equation 2.9, until it exceeds a predefined threshold probability, $p_i^{confirm}$, at which point it can be deduced that the target is actually located in cell i. In a similar manner, Equation 2.10 utilizes the lack of detection as the searcher passes over a given cell to update p_i, with the probability of target existence decreasing for that cell until falling below p_i^{deny}:

$$p\left(\begin{matrix}target\\present\end{matrix}\middle|\begin{matrix}\text{“}target\,not\\detected\text{”}\end{matrix}\right) = \frac{(\beta)p_i}{(1-\alpha)(1-p_i)+(\beta)p_i}. \tag{2.10}$$

For the purposes of this research, the target is assumed stationary; therefore, the overall probability that the target exists in the search area must remain at 100 percent. Thus, after the Bayesian update is accomplished for each individual cell, that change in probability must be distributed to all other cells in search area. In this way, the probability of target existence in cells where detections occur increases and in cells with no detections decreases, regardless of whether those cells have been physically searched yet. Table 2.3 summarizes the variables associated with the operations of the sensor.

2.3 The Nondeterministic Search Pattern

2.3.1 Development of the Levy Search

The Levy-based search utilized throughout this thesis is a trajectory of the searcher comprising piecewise linear segments of Levy-distributed lengths, as exhibited in numerous natural systems [18, 20, 21]. Such a model offers an initial mathematical basis for exploring the balance

Variable	Definition	Units
FOV	field of view of the UAV onboard sensor	[degrees]
alt	altitude of the searcher	[meters]
r_{sensor}	sensor operating range of detection	[meters]
A_{sensor}	circular area of sensor footprint	[meters^2]
α	sensor probability of false alarm	[none]
β	sensor probability of missed detection	[none]
p_i	current probability that a target is in a specific cell i	[none]
$p_i^{confirm}$	probability above which a grid cell is confirmed to contain a target	[none]
p_i^{deny}	probability below which a grid cell is assumed clear of a target	[none]

Table 2.3: Variables associated with sensor operations, as shown in Figure 2.6

between ideal search (for improved search effectiveness) and random search (for improved unpredictability in motion). We refer to the relationship of these attributes as the *efficacy* versus the *evasiveness* of the searcher, respectively. Upon initialization of the Levy search, the searcher is located at the starting position, (x_o, y_o), the Cartesian equivalent of the latitude and longitude representing the center of an area of uncertainty (AOU) in which it is desired to conduct the search. A random heading of the j^{th} iteration, γ_j, is drawn from a uniform distribution on turn angles from 0 to 2π. The individual length of the j^{th} flight leg, l_j, is drawn from the Levy distribution shown in Figure 1.1 and formulated (repeating Equation 1.2) as

$$L(l_j;\mu,c) = \sqrt{\frac{c}{2\pi}}\frac{e^{\frac{-c}{2(l_j-\mu)}}}{(l_j-\mu)^{\frac{3}{2}}}. \tag{2.11}$$

Recall that the scaling parameter, c, affects the uniformity of the distribution on lengths whereas the shift parameter, μ, limits the shortest length sampled from the distribution. The heading angle of each subsequent search leg, γ_{j+1}, is determined by adding a random heading change θ, again drawn from the uniform distribution from 0 to 2π, to the current searcher heading γ_j, as in Equation 2.12.

$$\gamma_{j+1} = \gamma_j + \theta, \qquad \text{with } \theta \sim U(0,2\pi). \tag{2.12}$$

The searcher travels from (x_o, y_o) to (x_f, y_f), the point located length l_j from the initial point and calculated as

$$x_f = x_o + l_j \cos(\gamma_j), \tag{2.13}$$

$$y_f = y_o + l_j \sin(\gamma_j). \tag{2.14}$$

A summary of these parameters is shown in Table 2.4 and illustrated by Figure 2.7.

Variable	**Definition**	**Units**
l_j	length of single flight leg j, drawn from $L(l_j; \mu, c)$	[meters]
γ_j	initial heading of searcher on leg j, drawn from $U(0, 2\pi)$	[radians]
θ	change of heading from leg j to leg $j+1$	[radians]
γ_{j+1}	heading on leg $j+1$, where $\gamma_{j+1} = \gamma_j + \theta$	[radians]
v	speed of searcher	[meters/sec]
r_{turn}	turn radius of the searcher	[meters]
t_{max}	maximum endurance of the searcher	[seconds]
(x_o, y_o)	Cartesian position of searcher at beginning of search leg j	
(x_f, y_f)	Cartesian position of searcher at end of search leg j	

Table 2.4: Variables associated with searcher characteristics

2.3.2 Bounded Levy Walk

The Levy walk has been shown to perform well while conducting a search over an infinitely large area due to the rapid dispersion from the search origin, resulting in larger area coverage over time [26]. In real-world operations, the search platform involved will have limited endurance capability and will generally be constrained to a finite area of operations defined by the AOU or FOC referenced in Section 2.2.1. To maximize the potential for the searcher to locate the target within these constraints, a reasonable estimate of target location must be known and bounds placed on this operational search area. This research explores two methods to confine the search pattern to the vicinity of the established AOU. The main intent of these methods is preserve searcher endurance by limiting the amount of excess coverage (i.e., coverage applied to areas outside of the AOU), but these bounding behaviors can also ensure the searcher is contained within the flight space cleared for the search, e.g., preventing a breach of territorial boundaries in contested areas. We introduce these methods here, and address them in detail in the following chapter.

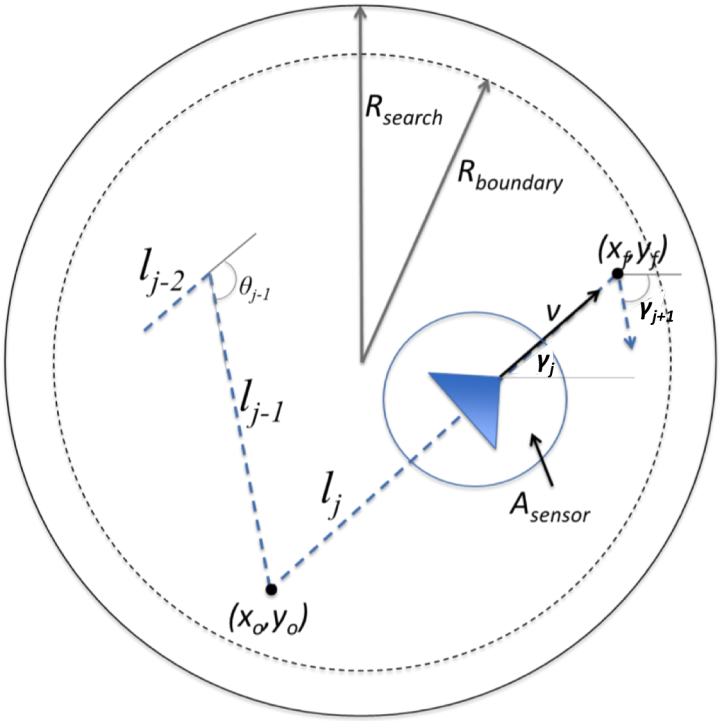


Figure 2.7: Illustration of the relevant parameters in determining the next search leg of the simulated Levy flight.

Turn-around

We refer to the first of the boundary enforcement methods as the "turn-around" method. As depicted in the basic description of the search pattern, a random leg length l_j is drawn from the Levy distribution function and applied to the flight path on a random heading α. After moving that leg of the search path, the "turn-around" algorithm compares the current searcher location to the center of the AOU. If it is determined that the searcher is beyond the bounds of the AOU, a bias is placed on the random headings, θ, ensuring the searcher moves back toward the AOU center. Once it is determined the searcher is back within the boundary of the search area the bias is lifted and the random search continues. This method allows the searcher to temporarily exit the search area, resulting in excess coverage and waste of precious endurance capacity and mission performance, but preserves the underlying Levy distribution for leg lengths.

Look-ahead

The second version is referred to as the "look-ahead" method. Through implementation of this method, a leg length and heading are drawn from the associated distributions and a calculation is performed, prior to travel, to determine the endpoint of the new leg. If the new heading and

angle combination places the searcher outside the bounds of the area, that l_j is discarded and both a new length and heading angle are drawn, repeatedly so until the sampling results in a terminal position that is in-bounds.

2.3.3 Dubins Levy Walk

The Levy-based search algorithm uses an instantaneous change of direction to transition from leg to leg. This is unrealistic when attempting to model the physical characteristics of a fixed wing UAV, which must constantly be moving forward at speed greater than the designed stall speed. We must account for the minimum turn radius capable by the UAV during a heading change in order to obtain a realistic portrayal of the actual flight path and area covered during the search. We model this realistic flight trajectory through the implementation of Dubins curves.

In 1957, Lester Dubins developed a system of equations to determine the optimal shortest path of a constant mover between any two waypoints, regardless of the heading configuration at each waypoint [28]. He derived that the path between these points could be broken down into three component parts: a starting turn, a straight-line path or second turn, and a finishing turn. The configuration of the three parts is based on the distance between the two points, the minimal turn radius of the mover, and the initial and final heading of the mover at the waypoints. By proving that each path can be broken down into these three path components, a total of six different paths exist. The set of these paths is known as Dubins set, which we annotate as $\mathbb{D}$. A visualization of these configurations is shown in Figure 2.8. Each of the turns is denoted by an *R* for right or *L* for left and the straight-line path is denoted as an *S*, such that the set of behaviors is:

$$\mathbb{D} = \{RSR, LSL, RSL, LSR, RLR, LRL\}. \tag{2.15}$$

In 2001, Shkel and Lumelsky analyzed the complex trigonometry conducted by Dubins and were able to categorize all Dubins paths between two points into Table 2.10, comprising sixteen configurations of initial and final mover headings [34]. By imposing a constraint on the mover that the distance between any two waypoints must be greater than four times the minimum turn radius of the mover, the authors ensure the turn circles never overlap (Figure 2.8[e] and [f]), eliminating the *LRL* and *RLR* sequences from the set of possible Dubins curve configurations. Depending on the quadrant associated with the initial and final angle headings (see Figure 2.9),there only exists up to four possible Dubins paths between the points, and in several cases, only one or two paths. These simplifications greatly reduce the number of calculations

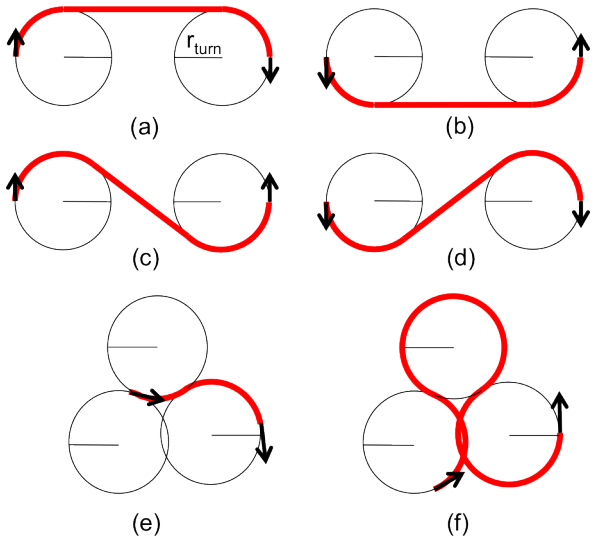


Figure 2.8: Graphical representation of the possible combinations of component parts of Dubins curves, illustrating trajectory configurations: (a) *RSR* (b) *LSL* (c) *RSL* (d) *LSR* (e) *RLR* (f) *LRL*.

required to find the optimal (shortest) path between the two waypoints.

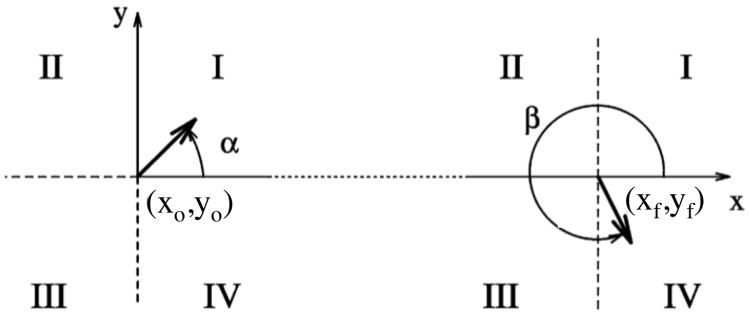


Figure 2.9: Shkel and Lumelsky determined that by limiting the distance between the two points to no less than four times the turn radius and determining the quadrant of the initial ($[x_o, y_o]$, α) and final ($[x_f, y_f]$,β) heading angles, the set of Dubins curves possible for each move is greatly simplified, derived from [34].

Final Quadrant / Initial Quadrant	1	2	3	4
1	RSL	if $S_{12} < 0$ then RSR if $S_{12} > 0$ then RSL	if $S_{13} < 0$ then RSR if $S_{13} > 0$ then LSR	if $S_{14}^1 > 0$ then LSR if $S_{14}^2 > 0$ then RSL else RSR
2	if $S_{21} < 0$ then LSL if $S_{21} > 0$ then RSL	if $S_{22}^1 < 0$ then LSL if $S_{22}^1 > 0$ then RSL if $S_{22}^2 < 0$ then RSR if $S_{22}^2 > 0$ then RSL	RSR	if $S_{24} < 0$ then RSR if $S_{24} > 0$ then RSL
3	if $S_{31} < 0$ then LSL if $S_{31} > 0$ then LSR	LSL	if $S_{33}^1 < 0$ then RSR if $S_{33}^1 > 0$ then LSR if $S_{33}^2 < 0$ then LSL if $S_{33}^2 > 0$ then LSR	if$S_{34} < 0$ then RSR if$S_{34} > 0$ then LSR
4	if $S_{41}^1 > 0$ then RSL if $S_{41}^2 > 0$ then LSR else LSL	if $S_{42} < 0$ then LSL if $S_{42} > 0$ then RSL	if $S_{43} < 0$ then LSL if $S_{43} > 0$ then LSR	LSR

Figure 2.10: Decision table developed by Shkel and Lumelsky for determining the shortest path between two points of known heading. From [34].

The initial and final heading configurations are determined using Figure 2.9. The distance calculations in [34] are conducted for each of the possible Dubins curves referenced from Table 2.10. The shortest of these distances represents the length, d,of a nominal Dubins path of configuration $D \in \mathbb{D}$, and is the sum of each of the three component parts; s_1^D, the arc length along the first turn, s_2^D, the length along the straight-line path, and s_3^D, the arc length along the second turn [34], such that:

$$d = s_1^D + s_2^D + s_3^D, \qquad D \in \mathbb{D}. \tag{2.16}$$

In this research, we further simplify the calculations of optimal path configuration by ensuring the heading of the searcher at the initial waypoint is always in the direction of the second waypoint, such that the relative heading between the two points is zero. This constraint reduces the determination of optimal Dubins path to be dependent only on the final heading of the searcher, eliminating the need for a *RSR* or *LSL* turn configuration. In summary, the constraints on heading and distance between the two points allows for the elimination of the *RSR*, *LSL*, *RLR*, and *LRL* configurations from the set of possible nominal paths. The searcher utilized in

this research only needs to conduct a *LSR* or *RSL* Dubins curve to achieve the nominal path between the two waypoints.

Formulation of Dubins Curves

This section provides a brief review of the geometric derivation of Dubins curves and calculation of their associated path lengths, so as to facilitate their implementation in the proposed simulation framework. A summary of all variables used in the calculation of Dubins curve lengths is shown in Table 2.5.As mentioned previously, only the *RSL* and *LSR* configurations are relevant to the presented study. We can begin with the computation of the path lengths for the *RSL* configuration. The geometry for calculating the Dubins path is shown in Figure 2.11.

Variable	**Definition**	**Units**
(x_o, y_o)	initial waypoint of search leg	[none]
(x_f, y_f)	final waypoint of search leg	[none]
γ_j	initial heading of searcher	[radians]
γ_{j+1}	final heading of searcher	[radians]
r	minimum turn radius of the searcher	[meters]
d	length of Dubins curve traveled between two waypoints	[meters]
D	configuration of Dubins curve, where $D \in \{RSL, LSR\}$	[none]
s_1^D	distance traveled during initial turn segment	[meters]
s_2^D	distance traveled during straight-line segment	[meters]
s_3^D	distance traveled during final turn segment	[meters]
(x_{ro}^D, y_{ro}^D)	pivot point (center) of turn circle for initial turn	[none]
(x_{rf}^D, y_{rf}^D)	pivot point (center) of turn circle for final turn	[none]

Table 2.5: Variables associated with the calculation of Dubins curves and their path lengths.

The searcher is assumed to have the same fixed turn radius for every heading change. This results in the straight-line path being along a tangent line between two circles of equal radius. There are only four possible tangent lines between any two circles, and because these circles are of equal radius, the intersection of the inner tangents occurs at the midpoint between the two circle centers; thus, referring to Figure 2.11, we can use the distance between the two pivot points, $(x_{ro}^{RSL}, y_{ro}^{RSL})$ and $(x_{rf}^{RSL}, y_{rf}^{RSL})$, and the turn radius, r_{turn}, to determine the straight-line distance and the distance traveled along each curve.The following is a step-by-step derivation of the geometry associated with the determination of Dubins distance using a RSL configuration. The calculations for the *LSR* configuration are conducted in an analogous manner.

We first establish the initial and final turn pivot points, $(x_{ro}^{RSL}, y_{ro}^{RSL})$ and $(x_{rf}^{RSL}, y_{rf}^{RSL})$, as the offset of π radians from the heading of the searcher at each point, at a distance of r_{turn} from the Levy

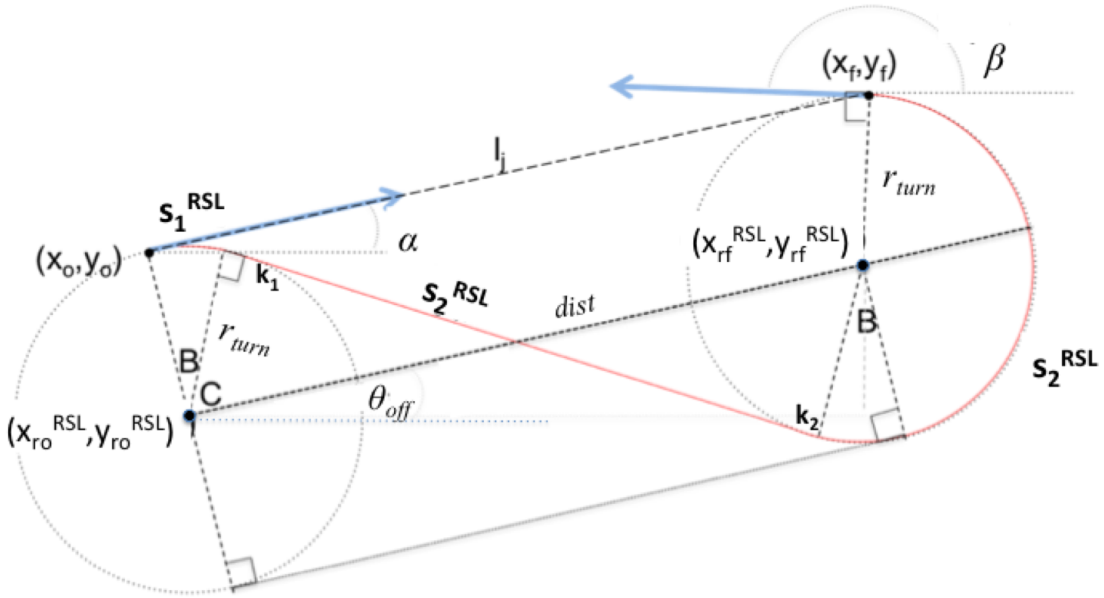


Figure 2.11: Illustration showing the variables required in the calculation of the Dubins distance. This example shows the geometry for a *RSL* configuration, but the process is similar for *LSR*.

waypoint (x_o, y_o) or (x_f, y_f), as shown in Figure 2.12. Having established the pivot points based on the Dubins configuration being traveled, we can now determine the distance, *dist*, and the offset,θ_{off}, between those pivots points for use in calculating the initial turn distance, s_1^{RSL}, and the straight-line travel distance, s_2^{RSL}.

$$dist = \sqrt{(x_{rf}^{RSL} - x_{ro}^{RSL})^2 + (y_{rf}^{RSL} - y_{ro}^{RSL})^2}$$

$$\theta_{off} = \sin^{-1}\left(\frac{y_{rf}^{RSL} - y_{ro}^{RSL}}{dist}\right)$$

We next calculate the angle, C, between the intersection point of s_2^D and the segment of length *dist*, and the tangent point of s_2^D on the first turn circle (k_1 in Figure 2.11).

$$C = \cos^{-1}\left(\frac{r_{turn}}{\frac{dist}{2}}\right)$$

The searcher is assumed to maneuver with a constant turn radius for both the initial and final turns, therefore, the angle between tangent point k_1 and the pivot point of the final turn is always $\frac{\pi}{2}$. Having calculated the value of angles θ_{off} and C, we can now calculate B, the angle

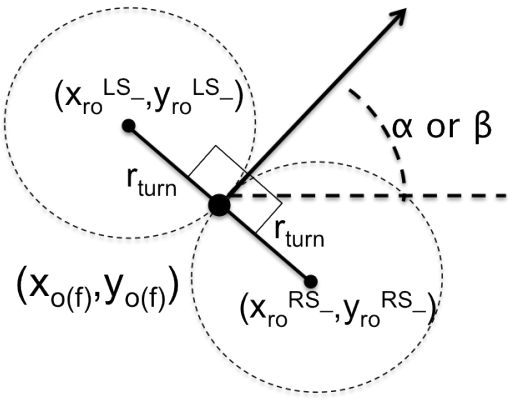


Figure 2.12: Illustration showing the configuration of pivot points at the initial and final waypoints of a search leg. The location of the pivot point is at a distance r_{turn} from the waypoint, offset π radians in the appropriate direction.

associated with the arc length of s_1^{RSL}, as

$$B = \frac{\pi}{2} - C - \theta_{off}$$

with the distance traveled in the initial turn simply being the length of the arc formed between the initial waypoint (x_o, y_o) and the tangent point k_1, or

$$s_1^{RSL} = Br_{turn}. \tag{2.17}$$

By a simple application of the Pythagorean theorem we calculate the straight-line distance as

$$s_2^{RSL} = 2\sqrt{\left(\frac{dist}{2}\right)^2 - r^2}. \tag{2.18}$$

The distance traveled during the second turn depends specifically on the desired final heading of the searcher, β. Again, because r_{turn} is equal for both initial and final turns, symmetry exists within the geometry of the Dubins path. In particular the angle, B, and arc length, s_1^{RSL}, between the tangent point and point perpendicular to the pivot are of equal value in both turns, simplifying the calculation of s_3^{RSL}.

$$s_3^{RSL} = \begin{cases} s_1^{RSL} + r_{turn}\beta, & \text{if } 0 \leq \beta \leq \frac{\pi}{2} \\ s_1^{RSL} + r_{turn}\left[\frac{\pi}{2} + \mod\left(\beta - \frac{\pi}{2}, 2\pi\right)\right], & \text{if } \frac{\pi}{2} < \beta \leq \pi \end{cases} \tag{2.19}$$

The distance traveled along the Dubins path is then the sum of the three component parts.

$$d^{RSL} = s_1^{RSL} + s_2^{RSL} + s_3^{RSL}.$$

The calculations for the *LSR* configuration are conducted in an analogous manner. Note that due to the turns occurring in opposite directions from the *RSL*, the ranges of β are different. The resulting path length, $d^{LSR} = s_1^{LSR} + s_2^{LSR} + s_3^{LSR}$, for the *LSR* Dubins path is summarized as:

$$s_1^{LSR} = Br_{turn}, \tag{2.20}$$

$$s_2^{LSR} = 2\sqrt{\left(\frac{dist}{2}\right)^2 - r_{turn}^2}, \tag{2.21}$$

$$s_3^{LSR} = \begin{cases} s_1^{RSL} + r_{turn}\left[\frac{\pi}{2} + \sin^{-1}\left(\frac{y_{rf}^{LSR} - y_f}{r_{turn}}\right)\right], & \text{if } \pi \leq \beta \leq \frac{3\pi}{2} \\ s_1^{RSL} + r\left[2\pi - \beta\right], & \text{if } \frac{3\pi}{2} < \beta \leq 2\pi \end{cases} \tag{2.22}$$

The Dubins distance traveled between the waypoints accounts for the constraints on turn radius observed in real-world flight, and results in a longer duration of travel than in the instantaneous Levy search where no turn radius is implemented. It is necessary to account for this difference in distance to ensure accurate calculation of time of flight, ultimately relating to an accurate depiction of time to target detection, time to counter-targeting of the searcher, and coverage rate. The implementation of the distance calculations in simulation is discussed in Chapter 3.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Modeling and Simulation

The flight pattern modeled in this research is a constant speed trajectory comprising segment lengths drawn from a Levy distribution. Such a model offers an initial mathematical basis for exploring the balance between linear ideal search (for improved search effectiveness) and random search (for improved unpredictability in motion). This chapter details the functional process of the computer simulation developed for this research. The general flow of the algorithm is represented by the block diagram in Figure 3.1and described in pseudo-code in Algorithm 3.1. The assumptions applied during the simulation and a more detailed description of each supporting function follow.

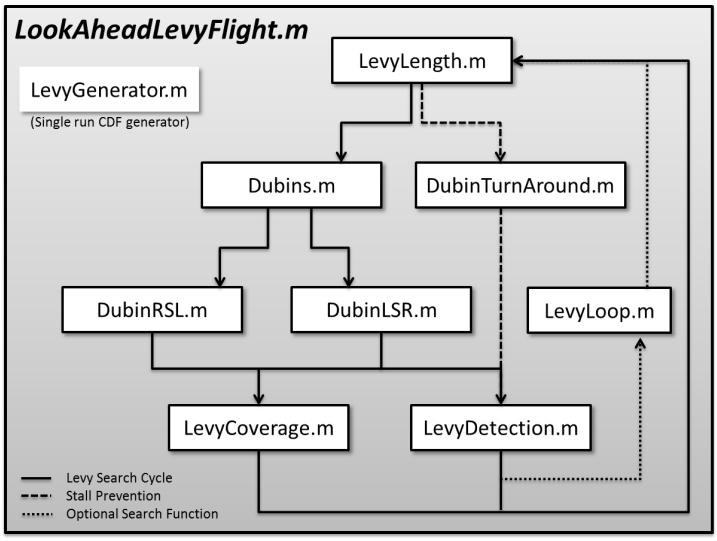


Figure 3.1: Block diagram showing the relationship and ordering of the various Matlab functions utilized throughout execution of the LookAheadLevyFlight.m script.

3.1 Simulation Model Assumptions

Although effort is taken to ensure realistic simulation parameters, several assumptions are applied to the model, distinguishing the simplistic dynamics of this simulation from the employment of a physical searcher in a real-world environment.

The first assumption is that the searcher maintains a constant altitude and speed throughout execution of the flight pattern. The sensor footprint at any discrete time step is based on the

altitude of the searcher (c.f. Equation 2.5), and the area coverage with respect to time is a function of searcher speed, as expressed in Equation 2.2. Holding these parameters constant at values representative of the average speed and altitude exhibited during real-world operations simplifies the coverage calculations and facilitates a dependence on elapsed flight time only.

The next assumption is that the sensor utilized by the searcher is mounted on some form of gimbal device. Most fixed-wing aircraft require banking while turning, which may result in skewing the shape and placement of a fixed sensor footprint. The realistic assumption of a gimbal device which keeps the sensor pointed directly below the searcher at all times eliminates the need for complicated calculations to determine actual sensor coverage shape and placement.

As discussed in detail in Chapter 2, the search grid is divided into one meter by one meter discrete cells. These grid cells are small relative to the overall search area; thus any grid cell that is located even partially within the sensor footprint is considered searched for the purposes of this research. This simplification alleviates the need to perform calculations on the exact flight path geometry to determine and track how much of an individual grid cell is searched during each pass of the searcher.

The search area is circular in geometry and is assumed to contain no obstructions which prohibit the flight of the searcher at its assigned altitude. In addition to supporting the assumption of constant altitude, this eliminates the need to model or discuss command and control connectivity issues between the searcher and its controlling station, which may otherwise arise due to hindrances in line of sight.

Each iteration of the flight pattern begins at the center of the circular search area; the transit from "home base" to the search area and back are not explicitly accounted for in this model. The search area is assumed to be within reasonable transit distance from the controlling station and does not affect the relative endurance of the UAV. What is not modeled is the UAV's susceptibility to counter-targeting during transit between the "home base" and the area of operations.

3.2 Developing the Levy Distribution

The use of Matlab facilitates existing access to common parametric distribution functions, such as the uniform and exponential distributions. Unfortunately, the Levy distribution is not among the list of common distributions provided in Matlab, and thus we can utilize the cumulative distribution function of the Levy, given by Equation 3.1 [45]., to generate an array of search leg lengths which can be accessed during the simulation. Pantaleo et al. discuss four different

methods for parameterizing the Levy distribution for use in simulation, of which we choose the transformation method to quantify the cumulative distribution function from the closed form Levy defined in Section 1.3 [25, 26, 45].

$$\begin{aligned} F(x;\mu,c) &= \int_{-\infty}^{x} L(l;\mu,c)\,dl, \\ &= erfc\left(\sqrt{\frac{c}{2(x-\mu)}}\right), \end{aligned} \tag{3.1}$$

where $erfc(x)$ is the complementary error function, defined as

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt.$$

There are an infinite number of lengths which can be returned from the Levy distribution depending on the accuracy to which we want the lengths represented. For simplicity we choose to represent each length to the whole meter (1.0 m). More precise measurements of leg lengths were attempted, up to 0.01 meters, but resulted in excessive simulation run times and memory use, e.g. the Levy CDF using 1 m precision took 0.28 seconds to build, whereas the 0.01 m precision CDF took 27.8 seconds, and 0.001 m precision 276.4 seconds. The one meter precision coincides with the discretization of the search area and, due to the relative size of the search area, provides an accurate representation of coverage ability. We limit the searcher to operations within the circular search area, and thus a distance greater than twice the radius of that area is never traveled on a single straight-line leg, which represents an upper bound on possible leg lengths. To address a lower bound on feasible leg lengths, we note that the assumed implementation of Dubins curves to simulate realistic flight physics requires waypoints separated by a minimum of four times the turn radius, although, in general, this minimum distance can be set to zero [34]. We set the value of the shift parameter, μ, to this value to represent this minimum distance. Although the transform algorithm incorporated here allows for the storage of varying degrees of precision on Levy lengths, we calculate only the cumulative probabilities of lengths within the defined range bound by the rounded values of μ to $2R_{search}$, to limit the run time and memory usage of the simulation when building the Levy CDF.

We apply each of these values to the Levy CDF by inputting an array, x, of lengths from μ to R_{search}, the scaling parameter, c, and the shift parameter,μ, to the **LevyGenerator.m** function and store the results in a CDF array. Each array index of the Levy CDF maps directly to the

index of the original x array of lengths used to build the distribution. To draw a random Levy length, a random variable u is drawn from a uniform distribution of zero to one, such that $u = U(0,1)$. The Levy CDF array is searched for this value u, and the index associated with the value in the CDF closest to u is returned, representing the index in the original x array containing the random Levy length, l_j. This process is referred to as inverse transform sampling [46].

Figure 3.2 shows the results of this inverse transform sampling for various values of the scaling parameter, c, as compared to the known graph of the Levy PDF. This method provides an accurate distribution on the Levy lengths for use in the simulation.

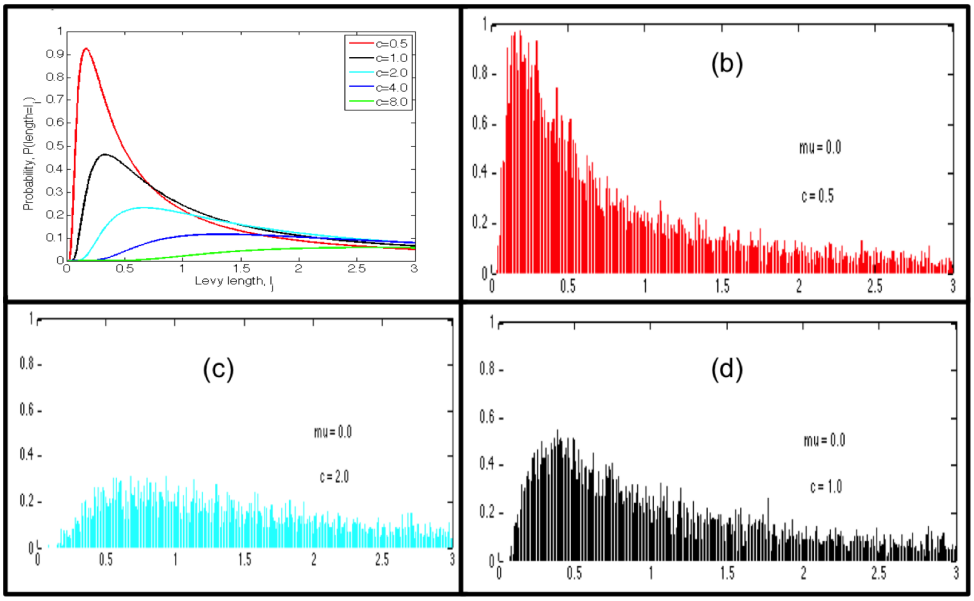


Figure 3.2: Comparison of the known shape of the Levy PDF (a) with the results of the inverse transform sampling technique implemented in Matlab for various scaling and shift parameters: (b) $\mu = 0$, $c = 0.5$, (c) $\mu = 0$, $c = 2.0$, (d) $\mu = 0$, $c = 1.0$

3.3 Restriction of the Searcher to a Finite Area

The searcher starts each leg of the flight at (x_o, y_o), initially assumed the center of the search area. With the length of the search leg established, a simple polar distance calculation is conducted to determine the final waypoint (x_f, y_f) on the current heading α per Equations 2.13 and 2.14. If (x_f, y_f) is within the boundary of the circular search area, the searcher travels along that leg. However, if traveling this (l_j, α) combination results in a violation of the area boundary (i.e., outside $R_{boundary}$, c.f. Figure 2.7), an adjustment is made to the trajectory. We propose and investigate two methods for conducting this adjustment.

Turn-around Method

We refer to the first of the two heuristics as the *turn-around* method. The searcher travels the entire length of each randomly drawn Levy leg regardless of the area bounds, which may result in the searcher temporarily exiting the area. In this adaptive algorithm, the searcher's position with respect to the area boundary is inspected at each waypoint and, upon recognition of an out-of-bounds condition, directs the searcher back towards the search area via a biased distribution on the turn angle, θ. The search area is divided into four quadrants equivalent to that of the Cartesian plane. Dependent upon which quadrant the searcher exceeds the search boundary, the limits of the uniform angle distribution are collapsed to ensure that the next heading places the searcher on a trajectory within ± 45 degrees of the center of the search area. This artificial bias on the heading angle is maintained until the searcher is recognized to be within the bounds of the search area. A breakdown of the algorithm is shown in Algorithm 3.2.

By applying the bias to the heading angle, no restriction is imposed on the lengths of the flight legs; thus the distribution of lengths in the flight path remains that of Equation 2.11. Preliminary investigation shows that although the case-dependent heading bias ensures the searcher returns to the finite area, the time spent outside of the search area cannot be easily predicted. This results in large segments of search time where no additional area coverage occurs, in effect wasting the searcher's valuable flight time. Given this operational constraint of limited endurance time in practical search missions, we choose to disregard the *turn-around* method for the remainder of this study.

Look-ahead Method

We refer to the second method for addressing the bounded search area as the *look-ahead* method. The searcher again starts at the center of the search area with random leg lengths drawn from Equation 2.11 and random heading changes from the uniform distribution on θ. Prior to completing a search leg, the length and angle of the subsequent leg are drawn and a calculation conducted to ensure that transit from the end of the current search leg (l_j) to the end of the subsequent search leg (l_{j+1}) will not place the searcher outside of the area boundary. If this new length/angle combination results in a violation of the boundary by the searcher, the length and angle are simply discarded and a new length and angle drawn. This process is repeated until achieving a length/angle pair which meets the confinement requirements of the area. This procedure is outlined in Algorithm 3.3.

The discarding of the lengths that would otherwise place the searcher outside the area boundary

appears to modify the original probability distribution. The histogram of the leg lengths actually traveled during execution of the *look-ahead* method is shown in Figure 3.3.

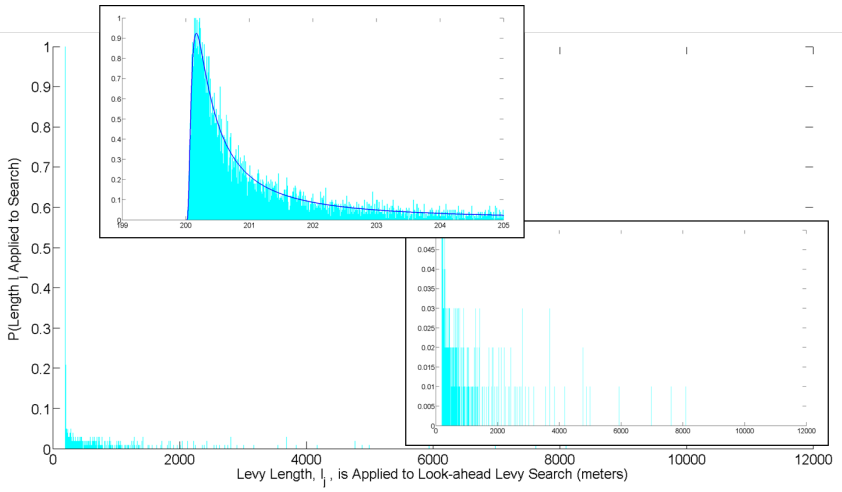


Figure 3.3: The probability histogram of Levy lengths applied to the *look-ahead* Levy search. The upper inlay magnifies the lower values of l_j to show the conformity to the Levy distribution. The lower inlay magnifies the lower probability region, showing that the Levy tail is effectively cutoff for values near twice the search radius, $R_{search} = 6000$.

We plot the CDF of the modified *look-ahead* Levy distribution versus the standard Levy distribution of Equation 3.1 and see that indeed, the containment process results in a more rapid rise to maximum probability. The histogram of the *look-ahead* simulation runs in Figure 3.3(b) shows that the general shape of the Levy is maintained, although there is a noticeable increase in probability for the legs less than $2R_{search}$, thus we enact the *look-ahead* containment method throughout this research, with future efforts devoted to examining the effects of this probability shift. The *look-ahead* method preserves precious search endurance as compared to the *turn-around* method of containment, however the *look-ahead* is not without fault. The potential exists for the searcher to require an infinite number of draws as the length and heading distributions are discarded due to violation of the boundary conditions. In real world operations this could result in inadvertent violation of the boundary by a "runaway searcher" with no updated course vector. As shown in Figure 3.1, we prevent this through implementation of a "stall prevention" *turn-around* function, **DubinsTurnAround.m**, after 500 consecutive unsuccessful length/angle combinations are drawn. This function simply implements a $\frac{5\pi}{6}$, or 150 degree, change in heading angle along the minimum turn radius of the searcher. This value was chosen to prevent the searcher becoming stuck in an infinite loop of 180 degree turns, representing a "fail-safe" for real-world UAV operations.

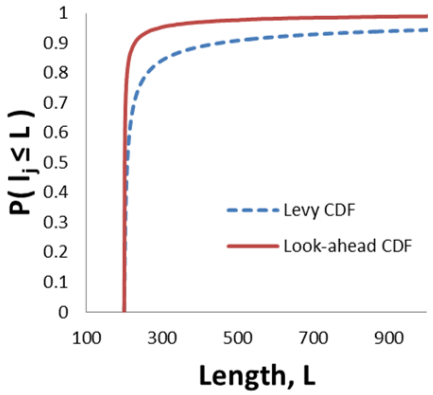


Figure 3.4: A plot of the CDF compiled from the lengths of the *look-ahead* Levy (solid) simulation versus the standard Levy CDF (dashed) demonstrates the probability shift to shorter lengths.

3.4 Algorithm for Searcher Movement

3.4.1 Discrete Sensor Sampling

With the search leg vector determined through the method above, the movement of the searcher is implemented in simulation through discrete steps. The searcher position is updated every r_{sensor} meters, where r_{sensor} is the search radius of the sensor. After each position update, the **LevyDetection.m** and **LevyCoverage.m** functions are run to determine the current operating picture. Using this discrete method to determine the amount of area covered accounts for 96% of the area that would have been covered through use of a continuous sweep, but saves a significant amount of simulation run time and processing power, as shown in Figure 3.5. A detailed derivation of the calculations used to determine this coverage value can be found in Appendix D.

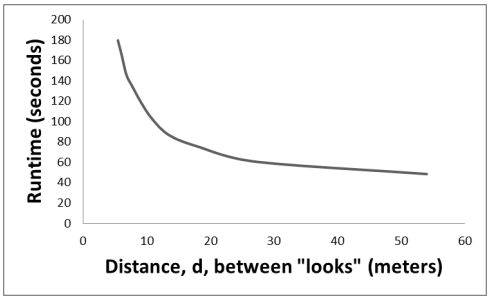


Figure 3.5: This graph shows the effects of varying the spacing between the discretized position updates on the simulation runtime. As the spacing increases, the simulation runtime decreases exponentially. By choosing a discretized spacing equal to the search radius, we achieve 96% of the coverage attained through a continuous search for a fraction of the runtime cost.

3.4.2 Movement Along the Flight Path

The implementation of Dubins curves is an important aspect to modeling real-world flight physics throughout this simulation. Once a leg length, l_j, is defined that meets the criteria set forth in Section 2.3.2, the initial and final waypoints and headings are passed to the **Dubins.m** function. Calculations are performed within this function to determine the optimal Dubins path creating the shortest distance between (x_o, y_o) and (x_f, y_f). The mathematical derivation of the Dubins calculations is addressed in Section 2.3.3 and the sequence of the **Dubins.m** function is described in Algorithm 3.4.

DubinsLSR.m and **DubinsRSL.m** calculate the discretized waypoints on each flight leg. The initial and final waypoints, (x_o, y_o) and (x_f, y_f), respectively, are passed to the appropriate function along with the segment lengths of the optimal path configuration as determined in **Dubins.m**. Depending on the path configuration, **DubinsLSR.m** or **DubinsRSL.m** generates an array of intermediate waypoints along the Dubins path, each with a spacing of r_{sensor} between, as discussed in Section 3.4.1.

Once the waypoint array is returned to the **LookAheadLevyFlight.m** via **Dubins.m** and **DubinsLSR.m** or **DubinRSL.m**, the algorithm begins iterating the array and plotting the waypoints. The flight time of the searcher is incremented after every intermediate waypoint using the distance between those waypoints, $\triangle d$ and the searcher speed, where

$$t_{flight} = t_{flight} + \frac{\triangle d}{v}. \tag{3.2}$$

As discussed in Section 3.4.1, we have defined the distance to be r_{sensor}. However, we use $\triangle d$ here to account for instances where the distance remaining on a given leg is less than r_{sensor}, as accounted for in Algorithm 3.5.

In Chapter 2, we introduced T_C, or the time the searcher can maneuver in the area before being counter-targeted by the adversary. As stated, this time is related to intelligence assessments of the technological capability, notably the time the enemy must maintain a target lock on the searcher prior to realizing an adequate firing solution. For simplicity, we have assumed that this target lock must be achieved during a straight-line path of the searcher; any change in speed or heading results in the need to restart targeting efforts. This time to target lock, T_{LOCK}, can thus be converted into a critical distance, $l_{crit} = T_{LOCK}v$, dependent on searcher speed which, if exceeded, results in the searcher being counter-targeted.

As the algorithm computes the waypoints, the total straight leg distance traveled on the flight leg is tracked. If this total straight leg distance exceeds l_{crit}, the searcher is considered counter-targeted, resulting in a mission failure. The flight time up to that waypoint is recorded as the time to counter-targeting, T_C, and the simulation is ended.

For each of the discretized leg waypoints, the existence of the target within the sensor radius is checked and an update to the overall area coverage is performed. Upon reaching the last coordinate in the waypoint array, a new length l_j and heading change θ are drawn and the process is repeated until target detection occurs, the searcher is counter-targeted, or the UAV reaches maximum flight time.

3.4.3 Levy Detection

The **LevyDetection.m** function is divided into two algorithms, one for conducting a Bayesian updated search of the area, the other for conducting the search through employment of a perfect cookie cutter sensor. In the case of the Bayesian search, each sensor "look" into a grid cell results in a Bayesian update as discussed in Section 2.2.2. The sum of the probability mass falling within the sensor footprint is updated according to Equations 2.9 and 2.10. The change in probability resulting from this *a posteriori* update is dispersed throughout the entire search area, as changes in probability proportional to the existing, or *a priori*, probability in each cell. Based on this update, a target detection occurs when the *a posteriori* probability in a cell exceeds a detection confirmation threshold, $p_c^{confirm}$. Over the duration of the search, the concentration of probability mass is "pushed" to the vicinity of the target location until, eventually, the probability of a single grid cell exceeds $p_c^{confirm}$ and the search ends in a successful detection. The Bayesian algorithm is also designed to account for sensor imperfection, such as probability of missed detection, β, and probability of false alarm, α. The less perfect the sensor, as indicated by higher values of α and β, the longer the time to target detection tends to be. The summary process of the Bayesian update on detection is shown in Algorithm 3.6.

The second algorithm in the **LevyDetection.m** function is a perfect detection model to focus the analysis on the search pattern itself, shown in Algorithm 3.7. In this algorithm, the target location is continually compared to the current searcher location. If the distance between target and searcher is less than the sensor range, r_{sensor}, the target is detected. The user specifies the type of detection algorithm to be used during the simulation initialization.

3.4.4 Levy Loop

In conjunction with the Bayesian algorithm of the Levy detection function, the Levy loop function is an optional algorithm intended to decrease the time to target detection and complete area coverage, which is implemented following the **LevyDetection.m** function, as shown in Figure 3.1. This function is **LevyLoop.m** divides the updated probability matrix, output from the Bayesian algorithm in **LevyDetection.m**, into equal sectors of size N by N cells, where N is defined by the user during simulation initialization. Each of these sectors represents a lower resolution probability mass function (computationally represented as a matrix) over the entire search area.

If no target detection occurs prior to the expiration of a preset loop timer, which begins counting down at the commencement of the search, a calculation is performed to determine which of the sectors contains the highest probability mass. This highest probability sector becomes the target destination for the searcher, with heading angles biased to force the searcher to the area, as shown in Algorithm 3.8. When the algorithm determines that the searcher is within the vicinity of the high probability area, as determined by the searcher crossing either the horizontal or vertical bounds of the sector, the loop timer is reset and the searcher resumes the *look-ahead* search.

3.4.5 Levy Coverage

The **LevyCoverage.m** function takes as an input the current waypoint of the searcher, just as in the **LevyDetection.m** function. The function maintains a $R_{\hat{search}} \times R_{\hat{search}}$ coverage matrix, where $R_{\hat{search}}$ is the whole number representation of the search area radius; the discretized search area. Initially containing all "zeros", each matrix block is flagged with a "one" if the sensor footprint of the searcher reaches the representative coordinates on the search area. It is assumed the sensor footprint is circular; thus only the matrix cells falling within a distance r_{sensor} from the searcher position are marked. For each waypoint, the number of ones in the matrix is totaled and divided by the total number of cells falling within the A_{search}, resulting in the coverage ratio at the current time of flight, as described in Algorithm 3.4.5.

$$F(t) = \frac{\text{\# matrix cells flagged at } t_{flight}}{A_{search}} \tag{3.3}$$

3.5 Graphical Display

The simulation is equipped with a graphical display that may be turned on or off as the user desires. The display, a screen shot of which is shown in Figure 3.6, shows the current flight trajectory of the searcher, a warning meter to indicate if the searcher is in danger of exceeding the critical length, l_{crit}, and a plot of coverage ratio versus flight time. The flight trajectory plot displays the circular search area as a black line, and the boundary area as a dotted black line. The target location (unknown to the searcher) is denoted by an asterisk. The searcher is plotted as a dark square surrounded by a lighter colored circle representing the sensor footprint. As the searcher moves throughout the area the flight path history and area covered are maintained on the plot.

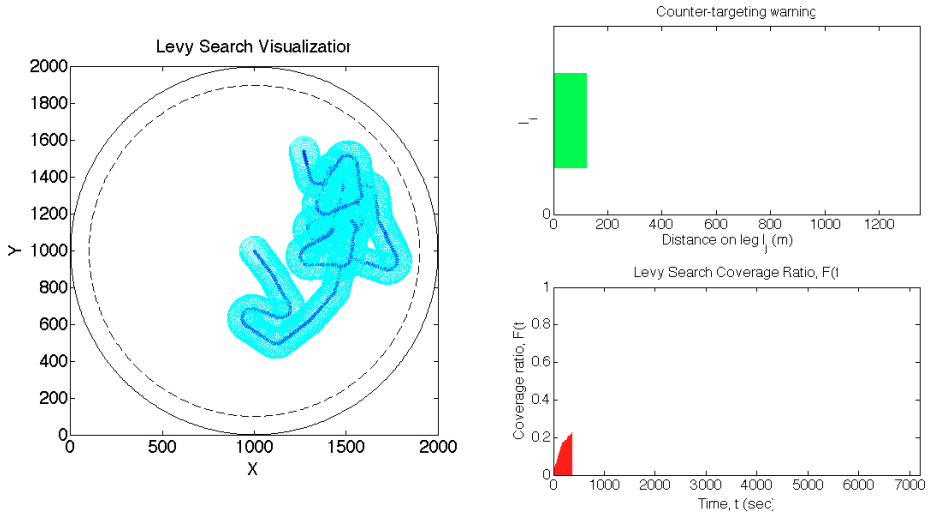


Figure 3.6: Graphical display showing search pattern, coverage ratio, and a warning meter for proximity to counter-targeting.

The l_{crit} warning meter in the upper right of the display gives a visual representation of the probability of the searcher being counter-targeted. As the searcher travels down the straight-leg path the meter begins to increase, as represented by a green horizontal bar graph which plots the distance traveled on that leg in meters. As the distance traveled exceeds 75% of l_{crit}, the meter turns yellow. There is no interactive functionality to maneuver the searcher, but it does give the user a visual alert that the searcher may be on the verge of counter-targeting. If the straight-leg distance exceeds l_{crit}, then the meter turns red and the simulation ends.

The bottom right of the display houses the plot of coverage ratio versus flight time. Each time

LevyCoverage.m is run, the computed $F(t)$ is plotted on this graph against the current time of flight, in seconds.

This simulation model encompasses both a visual and mathematical representation of nondeterministic search performance. The graphical output can give the user a feel for the type of coverage patterns evolving from such a search, whereas the mathematical output on times to target detection and counter-targeting help establish a quantifiable measure on searcher performance. In the next chapter we conduct cursory analysis on these measures of performance to demonstrate the capabilities inherent to the simulation model.

Algorithm 3.1 LookAheadLevyFlight.m

Build the Levy CDF using **LevyGenerator.m**
Initialize searcher position (x_o, y_o) and heading α, target position, critical length, and simulation time
while Flight time < t_{max} **do**
  Draw l_j from Levy CDF using **LevyLength.m** and heading change θ from $U(0, 2\pi)$
  Calculate search leg endpoint (x_f, y_f) and final heading $\beta = \alpha + \theta$
  if (x_f, y_f) does not exceed search area boundary **then**
    Determine optimal path from (x_o, y_o) to (x_f, y_f) using **Dubins.m**
    while Waypoints remain on current leg **do**
      if Straight-line leg distance exceeds critical length **then**
        Searcher counter-targeted. *MISSION FAIL*
      end if
      if Target located via **LevyDetection.m then**
        Target detection. *MISSION SUCCESS*
      end if
      Update $F(t)$ using **LevyCoverage.m**
      Increment flight time for distance traveled and move to next waypoint on current leg
    end while
  else
    Determine turn around waypoints using **DubinsTurnAround.m**
    while Waypoints remain in turn **do**
      if Target located via **LevyDetection.m then**
        Target detection. *MISSION SUCCESS*
      end if
      Update $F(t)$ using **LevyCoverage.m**
      Move to next waypoint in turn
      Increment flight time for distance traveled and move to next waypoint on current leg
    end while
    $\alpha = \alpha + \frac{5\pi}{6}$
  end if
  $\alpha = \beta$
  $x_o = x_f$
  $y_o = y_f$
end while

Algorithm 3.2 Levy search using the *turn-around* method

Initialize searcher position (x_o, y_o) and heading α
while $t < t_{\max}$ **do**
  Draw l_j from Levy CDF using **LevyLength.m** and heading change θ
  Calculate search leg endpoint $\left(x_f, y_f\right)$ and final heading $\beta = \alpha + \theta$
  Move searcher to $\left(x_f, y_f\right)$
  if Searcher is outside of area boundary **then**
    Calculate heading ϕ from (x_f, y_f) to search area center
    Draw l_j from Levy CDF using **LevyLength.m**
    Draw θ from biased heading distribution $U(\frac{-\pi}{4}, \frac{\pi}{4})$
    Calculate search leg endpoint $\left(x_f, y_f\right)$ and final heading $\beta = \phi + \theta$
    Move to (x_f, y_f)
  end if
  Increment flight time for distance traveled and move to next waypoint on current leg
end while

Algorithm 3.3 Levy search using the *look-ahead* method

Initialize searcher position (x_o, y_o) and heading α
while $t < t_{\max}$ **do**
  Draw l_j from Levy CDF using **LevyLength.m** and heading change θ
  Calculate search leg endpoint $\left(x_f, y_f\right)$ and final heading $\beta = \alpha + \theta$
  while $x_f^2 + y_f^2 > R_{boundary}^2$ **do**
    Draw l_j from Levy CDF using **LevyLength.m** and heading change θ
    Calculate search leg endpoint $\left(x_f, y_f\right)$ and final heading $\beta = \alpha + \theta$
  end while
  Move searcher to (x_f, y_f)
  Increment flight time for distance traveled and move to next waypoint on current leg
end while

Algorithm 3.4 Dubins.m

Input: $l_j, (x_o, y_o), \alpha, (x_f, y_f), \beta$
if $\alpha < \beta \leq \alpha + \pi$ **then**
  Calculate $s_1^{RSL}, s_2^{RSL}, s_3^{RSL}$
  Determine the intermediate waypoints using **DubinsRSL.m**
else if $\alpha > \beta \geq \alpha - \pi$ **then**
  Calculate $s_1^{LSR}, s_2^{LSR}, s_3^{LSR}$
  Determine the intermediate waypoints using **DubinsLSR.m**
end if
Intermediate waypoints returned to **LookAheadLevyFlight.m** for use in plotting track and input to **LevyDetection.m** and **LevyCoverage.m**

Algorithm 3.5 DubinsLSR.m and DubinsRSL.m, where $\overline{D} \in \{LSR, RSL\}$

Input: $s_1^{\overline{D}}, s_2^{\overline{D}}, s_3^{\overline{D}}, (x_o, y_o), \alpha, (x_f, y_f), \beta, r_{sensor}$
Initialize segment distance traveled, *distance*, to zero
angle = $\alpha \pm \frac{\pi}{2}$, dependent on $\overline{D}$
Determine pivot point of initial turn, $(x_{ro}^{\overline{D}}, y_{ro}^{\overline{D}})$
while $distance < s_1^{\overline{D}}$ **do**
  if $s_1^{\overline{D}} - distance \geq r_{sensor}$ **then**
    Calculate waypoints along initial turn arc of radius r_{turn} at a spacing of r_{sensor}
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array for return to **Dubins.m**
    angle = angle $\pm \frac{r_{sensor}}{r_{turn}}$, dependent on $\overline{D}$
    $distance = distance + r_{sensor}$
  else
    Calculate final waypoint on initial turn arc at a spacing of $s_1^{\overline{D}} - distance$
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array for return to **Dubins.m**
    angle = angle $\pm \frac{s_1^{\overline{D}} - distance}{r_{turn}}$, dependent on $\overline{D}$
    $distance = distance + \left(s_1^{\overline{D}} - distance\right)$
  end if
end while
Initialize segment distance traveled, *distance*, to zero
angle = $\alpha \pm \frac{\pi}{2}$, dependent on $\overline{D}$
while $distance < s_2^{\overline{D}}$ **do**
  if $s_2^{\overline{D}} - distance \geq r_{sensor}$ **then**
    Calculate waypoints along straight-line segment of heading *angle* at a spacing of r_{sensor}
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array
    $distance = distance + r_{sensor}$
  else
    Calculate final waypoint on straight-line segment at a spacing of $s_2^{\overline{D}} - distance$
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array
  end if
end while
Initialize segment distance traveled, *distance*, to zero
Determine pivot point of final turn, $(x_{rf}^{\overline{D}}, y_{rf}^{\overline{D}})$
while $distance < s_3^{\overline{D}}$ **do**
  if $s_3^{\overline{D}} - distance \geq r_{sensor}$ **then**
    Calculate waypoints along final turn arc of radius r_{turn} at a spacing of r_{sensor}
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array
    angle = angle $\pm \frac{r_{sensor}}{r_{turn}}$, dependent on $\overline{D}$
    $distance = distance + r_{sensor}$
  else
    Calculate final waypoint on final turn arc at a spacing of $s_3 3^{\overline{D}} - distance$
    Store $(x_{waypoint}, y_{waypoint})$ in intermediate waypoint array
    angle = β
    $distance = distance + \left(s_3^{\overline{D}} - distance\right)$
  end if
end while
Return intermediate waypoint array to **Dubins.m**

Algorithm 3.6 Bayesian update algorithm incorporated in LevyDetection.m

Input: $prob\_density, \alpha, \beta, (x_{pos}, y_{pos}), r_{sensor}$
for Each cell c within r_{sensor} of (x_{pos}, y_{pos}) **do**
$p^{sensor}_{prior} = \sum_c p_c$
end for
if (Target < r_{sensor} from (x_{pos}, y_{pos}) AND random $U(0,1) > \beta$ (no missed detection)) OR (random $U(0,1) < \alpha$ (false detection)) **then**
Detection update : $p^{sensor}_{posterior} = \frac{(1-\beta)p^{sensor}_{prior}}{(\alpha)(1-p^{sensor}_{prior})+(1-\beta)p^{sensor}_{prior}}$
else
Nondetection update : $p^{sensor}_{posterior} = \frac{(\beta)p^{sensor}_{prior}}{(1-\alpha)(1-p^{sensor}_{prior})+(\beta)p^{sensor}_{prior}}$
end if
$\triangle p = p^{sensor}_{posterior} - p^{sensor}_{prior}$
for All cells c within $prob\_density$ **do**
$p^c_{posterior} = p^c_{prior} + p^c_{prior}\left(\frac{\triangle p}{p^{sensor}_{prior}}\right), \forall c$ within r_{sensor} of (x_{pos}, y_{pos})
$p^c_{posterior} = p^c_{prior} + p^c_{prior}\left(\frac{\triangle p}{p^{sensor}_{prior}}\right), \forall c$ beyond r_{sensor} from (x_{pos}, y_{pos})
if $p^c_{posterior} > p^c_{confirm}$ **then**
Target is detected
T_D = flight time
End simulation *MISSION SUCCESS*
end if
end for

Algorithm 3.7 Perfect cookie cutter sensor detection algorithm incorporated in LevyDetection.m

Searcher located at position (x_{pos}, y_{pos})
Target located at position (x_{target}, y_{target})
if Distance between (x_{target}, y_{target}) and (x_{pos}, y_{pos}) < r_{sensor} **then**
Target is detected
T_D = flight time
End simulation
end if

Algorithm 3.8 LevyLoop.m

Input: $sector\_prob, \alpha, (x_{pos}, y_{pos})$
Determine coordinates of search area sector containing highest probability mass, (x_{sector}, y_{sector})
Determine direction, γ, of (x_{sector}, y_{sector}) from current searcher position, (x_{pos}, y_{pos})
if $0 < \gamma \leq \frac{\pi}{2}$ **then**
  while $x_{pos} < x_{sector}$ OR $y_{pos} < y_{sector}$ **do**
    Draw l_j from LevyCDF using **LevyLength.m**
    Draw θ from biased distribution, $U(\theta - \frac{\pi}{9}, \theta + \frac{\pi}{9})$
    Move searcher distance l_j on heading θ
  end while
else if $\frac{\pi}{2} < \gamma \leq \pi$ **then**
  while $x_{pos} > x_{sector}$ OR $y_{pos} < y_{sector}$ **do**
    Draw l_j from LevyCDF using **LevyLength.m**
    Draw θ from biased distribution, $U(\theta - \frac{\pi}{9}, \theta + \frac{\pi}{9})$
    Move searcher distance l_j on heading θ
  end while
else if $\pi < \gamma \leq \frac{3\pi}{2}$ **then**
  while $x_{pos} > x_{sector}$ OR $y_{pos} > y_{sector}$ **do**
    Draw l_j from LevyCDF using **LevyLength.m**
    Draw θ from biased distribution, $U(\theta - \frac{\pi}{9}, \theta + \frac{\pi}{9})$
    Move searcher distance l_j on heading θ
  end while
else
  while $x_{pos} < x_{sector}$ OR $y_{pos} > y_{sector}$ **do**
    Draw l_j from LevyCDF using **LevyLength.m**
    Draw θ from biased distribution, $U(\theta - \frac{\pi}{9}, \theta + \frac{\pi}{9})$
    Move searcher distance l_j on heading θ
  end while
end if
Reset loop timer to t_{loop}
Resume search defined in **LookAheadLevyFlight.m**

Algorithm 3.9 LevyCoverage.m

Input: $coverage\_plot, (x_{pos}, y_{pos}), r_{sensor}$
Coverage cursor (x_{coord}, y_{coord}) set at coordinates $(x_{waypoint} - r_{sensor}, y_{waypoint} - r_{sensor})$
while $x_{coord} <= x_{pos} + r_{sensor}$ **do**
  $y_{coord} = y_{pos} - r_{sensor}$
  while $y_{coord} <= y_{pos} + r_{sensor}$ **do**
    if $\sqrt{(x_{pos} - x_{coord})^2 + (y_{pos} - y_{coord})^2)} \leq r_{sensor}$ **then**
      Position (x_{coord}, y_{coord}) flagged as searched on coverage matrix
    end if
    $y_{coord} = y_{coord} + 1$
  end while
  $x_{coord} = x_{coord} + 1$
end while
Area covered = $\frac{\text{\# of 1's in coverage matrix}}{\text{Total number of cells in search area}}$

CHAPTER 4:
Results, Analysis, and Experimentation

The purpose of this thesis is the development of a nondeterministic search pattern capable of producing area coverage comparable to that of the perfect continuous search, while maintaining a sufficient element of randomness conducive to avoiding counter-targeting of the searcher. This chapter discusses the results and analysis of simulations performed using various input parameters to the Levy distribution and area configuration to develop an understanding of the time to detection of a target, captured by the coverage ratio, as well as the time until counter-targeting, represented as the adversary's (unknown) capability.

We first investigate the properties of the search model via an expression for the Levy search coverage ratio, which is used to define describe the distribution on the time until the target is detected. The inclusion of the counter-targeting element allows for simulation and calculation of the mission success performance, which is explored through sensitivity analysis on the relevant parameters, such as the Levy scaling parameter, c, and the critical time until the searcher is counter-targeted, T_{LOCK}. Further discussion of the operational use of these insights is developed, presenting capability versus vulnerability curves to be used for decision support. To extend this work's operational relevance, additional study relaxes the assumption of perfect sensors by allowing for false positive and false negative detections via a Bayesian search framework. Finally, the chapter concludes with a description of live-fly field experiments conducted at Camp Roberts employing the proposed nondeterministic search trajectory generation approaches. Implementation using actual tactical UAVs in these actual experiments provide proof-of-concept validation of the presented methods.

4.1 Probability of Mission Success

Incorporating the primary objective of this research to investigate the impact of the counter-targeting capability of an adversary against a friendly searching UAV, we define a measure of performance referred to as the *probability of mission success*. Plainly stated, this metric quantifies the probability that the searcher finds the stationary target prior to being counter-targeted by an enemy weapon system. Recall from Section 2.1 that we refer to the time elapsed from the start of the search through target detection as T_D, and the time elapsed from the start of the search through the searcher being counter-targeted as T_C. The *probability of mission success*

is then represented as

$$P(T_D < T_C). \tag{4.1}$$

For the purpose of this thesis, we make a distinction between counter-targeting and counterdetection. We assume that the territory over which we are searching is under the influence of some anti-access/area denial (A2AD) tactic. In other words, the adversary is aware of the searcher's presence in the area (assumed counterdetection), and makes efforts to prevent the searcher from locating the target through the use of some anti-air asset (counter-targeting). This asset may be a conventional weapon, such as a man-portable air defense system (MANPADS), requiring unambiguous travel of the UAV in order for the user to obtain a sufficient target lock and effect a hard kill. As countries make technological advances in anti-UAV weaponry, the probability exists for increased use of directed energy weapons, requiring a finite amount of laser contact time with the surface of the UAV in order to overheat sensitive electronics and effect a mission kill. The capability to fire may even be limited by an adversary's doctrine, requiring a finite amount of time or specific number of maneuvers by the targeted UAV before a firing solution is considered valid. Recall also from Section 3.4.2, we make the assumption that this finite amount of time, annotated T_{LOCK}, is an intelligence assessment of enemy capability, and represents a random variable with a prescribed probability function. This T_{LOCK} is used to determine the maximum straight-line distance, l_{crit}, that can be traveled by the searcher prior to being counter-targeted.

4.2 Determining Coverage Ratio

Recall from Chapter 2 that the coverage ratio is the amount of search area that is uncovered by the searcher as a function of time. As discussed in Section 2.1, the continuous perfect search results in a linear increase to complete coverage, where as the continuous random search results in a logarithmic rise to complete coverage due to the propensity for repeated searches over the same grid cell. We compare the performance of the *look-ahead* Levy search developed in this research to these search pattern benchmarks. The discretization of the search area and the uniform distribution on the location of the single target allow us to use the coverage ratio as a measure of performance for the estimated time to target detection.

The simulation model is run using the searcher parameters shown in Table 4.1 to determine the estimated time required to search the entire area. The results of a single run of the simulation, illustrated in Figure 4.1, reveals that in the early stages of the search the coverage rate follows

that of the continuous ideal searcher and is nearly linear. Note that there are small increments of time which the search algorithm produces results better than what we have described as the "perfect" coverage. This is due to imperfections in the search algorithm caused by the discretization of the search space, and over time has minimal impact on the search results. As the search progresses, the probability of searching grid cells which have already been uncovered increases, and the effects of double coverage cause a more logarithmic shape to the curve, slowing the coverage rate.

Variable	Value	Units
v	25	[meters/sec]
c	0.5	[none]
μ	200	[meters]
R_{search}	6000	[meters]
r_{sensor}	161	[meters]

Table 4.1: Input parameters to simulation runs used to evaluate coverage rate.

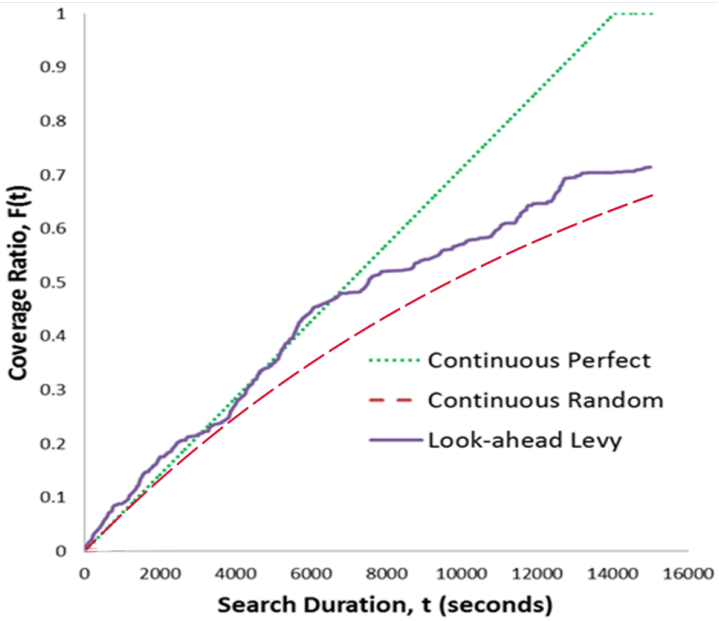


Figure 4.1: A single run of the simulation model shows that during the early stages of the search, a coverage rate similar to that of the continuous ideal searcher is achieved. However, as more area is uncovered, the randomness of the pattern causes the searcher to double back over areas previously searched, resulting in an logarithmic shape to the curve.

Given the absence of a closed-form expression for coverage ratio, $F(t)$, the simulation is executed $n = 1000$ times using the same parameters, with the objective of using regression analysis

to determine an expression for the (expected) area coverage as a function of time from these runs. We conduct a regression analysis on the data to construct a metamodel, denoted $\hat{F}(t)$, to approximate the coverage as a function of time. Noting that the shape of the expected coverate ratio mimics a logarithmic rise to complete coverage, similar to that of the random search described in Section 2.1, and so we choose the random search equation as the desired parametric form.

To conduct this analysis, we first transform the parametric coverage equation (Equation 2.3) into linear form to allow for use of Microsoft Excel's *Data Analysis* package.

$$F(t) = 1 - e^{-\lambda t}$$
$$-\log(1 - F(t)) = \lambda t$$

We then convert the coverage values obtained from the simulation runs into this form, by taking the natural logarithm of one minus the simulated values. A regression is run with this calculated log value as the dependent variable, and the simulated time observations as the independent variable. The regressor coefficients are shown in the final coverage ratio equation, Equation 4.2.

$$\hat{F}(t) = 1 - e^{-(0.0000813t + 0.035178)}. \tag{4.2}$$

The statistical results output from Microsoft Excel are summarized in Table 4.2, and the plot of the fitted regression values are plotted against the mean simulated coverage values in Figure 4.2.

The coefficients determined from the regression analysis result in a good statistical fit, as indicated by the p-value and R-squared values in the output file. We note that in the random search case, for the given parameter values, the equivalent coefficient (i.e., the coverage rate) is computed to be $\frac{vw}{A_{search}} = 7.12 \times 10^{-5}$, which is of the same order of magnitude as the fitted Levy coverage rate. However, attempts at further analysis of the regression coefficients does not offer insight as to the relation between the coverage ratio and the physical attributes of the searcher or search area. Further, using this expression for the coverage ratio as an estimate for time to target detection does not account for the possibility that the searcher is actively counter-targeted.

SUMMARY OUTPUT				0				
Regression Statistics								
Multiple R	0.999394							
R Square	0.998789							
Adjusted R Square	0.998789							
Standard Error	0.011409							
Observations	4641							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	498.0298	498.0298	3825968	0			
Residual	4639	0.603863	0.00013					
Total	4640	498.6336						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.035178	0.000336	104.5482	0	0.034519	0.035838	0.034519	0.035838
Time	8.13E-05	4.16E-08	1956.008	0	8.12E-05	8.14E-05	8.12E-05	8.14E-05

Table 4.2: Regression output from Excel *Data Analysis* tool. The low p-value and high R-squared values indicate the goodness of fit to the coverage rate.

4.3 Incorporating Counter-Targeting

In order to facilitate investigation of the impact of the adversary's ability to target the searcher, numerous simulation studies are performed that include variation on parameters relevant to both the Levy-based searcher and the enemy's targeting capability. For the initial studies, we consider values of searcher speed, v, and sensor sweep (half) width, r_{sensor}, to be comparable to those exhibited in an average mid-sized UAV flying at a 600 meter altitude with a 30° sensor field-of-view (c.f. Figure 2.6). Further, the Levy shift parameter, μ, is held constant and dictated by the simplifying assumptions of the Dubins algorithm discussed in Section 2.3.3. The target is uniformly randomly positioned in the circular search area.

Variable	**Value**	**Units**
T_{LOCK}	90	[seconds]
v	25	[meters/sec]
c	{ 0.2, 0.5, 1.0, 3.0 }	[none]
μ	200	[meters]
R_{search}	{3000, 6000, 12000}	[meters]
r_{sensor}	161	[meters]

Table 4.3: Input parameters to simulation runs measuring probability of mission success against an anti-UAV asset with a T_{LOCK} capability of 90 seconds.

Each execution of the simulation continues until one of three possible end states is reached: either the target is detected, the searcher is counter-targeted, or the endurance of the searcher

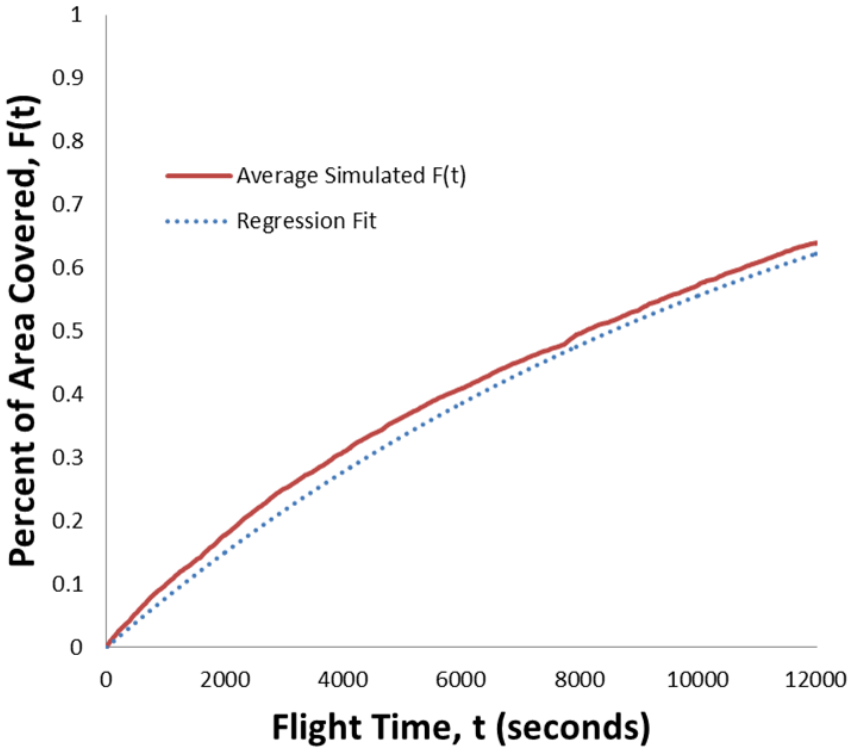


Figure 4.2: Plot of the mean coverage results from the $n = 1000$ simulation runs using the parameters in Table 4.1. The fitted regression curve is plotted for comparison.

times out. The first condition, a target detection, is recorded as a mission success, whereas the latter two conditions, either counter-targeting of the searcher or exceeding of the searcher endurance, registers a mission failure.

The initial numerical study explores variations in controllable parameters, namely those that could potentially be dictated by the UAV commander, to identify the impact of the size of the search area as well as the scaling parameter of the chosen Levy distribution. In particular, given previous works studying Levy flights on the infinite plane, the effect of the finite search area merits investigation. The time necessary to achieve target lock is held constant at $T_{LOCK} = 90$ seconds, which is a representative duration that assumes perfect intelligence of the adversary's anti-air targeting capability. The simulation model is run $n = 1,000$ times for each of several different parameter combinations, derived from the values shown in Table 4.3. The results of these simulations are summarized in Table 4.4.

Plots of the probability of mission success versus the search area size and magnitude of the

c	R_{search} [meters]	$\mathbb{E}[T_D]$ [minutes]	$\mathbb{E}[T_C]$ [minutes]	$P(Success)$
0.2	3000	56.2	65.5	0.53
0.2	6000	87.5	15.1	0.04
0.2	12000	23.3	26.4	0.02
0.5	3000	65.2	76.3	0.41
0.5	6000	62.2	39	0.03
0.5	12000	N/A	26.5	0.00
1.0	3000	34.1	41.1	0.40
1.0	6000	3.3	9.8	0.05
1.0	12000	16.9	14.8	0.01
3.0	3000	26.8	30.4	0.29
3.0	6000	4.5	7.5	0.07
3.0	12000	9.7	9.6	0.01

Table 4.4: Results of $n = 1000$ simulation runs against an anti-UAV asset with a T_{LOCK} of 90 seconds. Mission success is achieved upon target detection, with mission failure occurring if the searcher is counter-targeted.

scaling parameter are shown in Figure 4.3. These graphs indicate that as the search area radius, R_{search}, is increased for a given scaling parameter, c, there is an exponentially detrimental effect on the probability of mission success. In contrast, when T_{LOCK} is known and constant (though, in reality, this is a random variable), changes of c for a given R_{search} have much less effect on the probability of mission success. Based on these results, we focus the next series of simulations on the relationship between T_{LOCK} and R_{search} to identify the sensitivity of mission success on these parameters, which may provide further insights into the viability of future theoretical coverage models of Levy walks in finite areas.

We conduct $n = 100$ replications of each experimental design point, varying values for R_{search} and T_{LOCK}, with a fixed value of $c = 0.5$. The remaining search parameters are consistent with those in Table 4.3. The results of these simulations are presented in Table 4.5 and Figure 4.4.

The results of the simulations focusing on the relationship between R_{search} and T_{LOCK} again indicate that R_{search} has a more dramatic effect on the probability of mission success. A pairs plot of the data is shown in Figure 4.5. The first item to note on this plot is that as the critical length increases, meaning a longer duration of T_{LOCK}, the probability of mission success increases. An increase in T_{LOCK} simply means that the searcher could fly straighter longer (i.e., l_{crit} is longer), which was, for this set of simulations, controlled by holding the Levy scaling parameter constant for a consistent probability distribution on the Levy search legs.

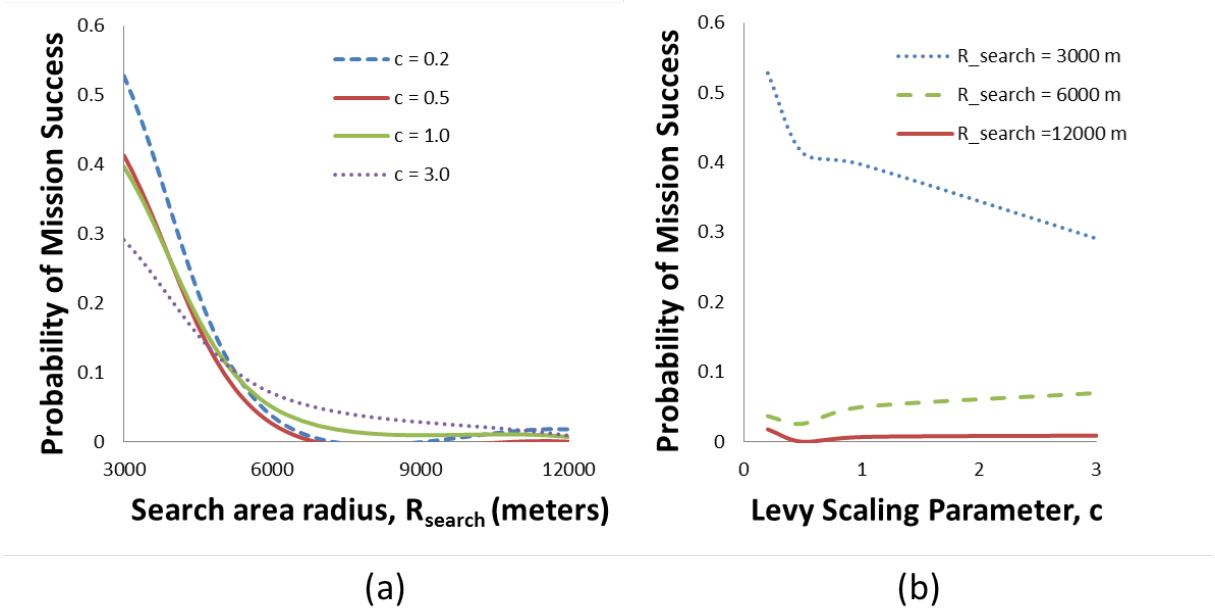


Figure 4.3: (a) Plot showing the effect on mission success of an increase in R_{search}, given scaling parameter, c, and T_{LOCK}. (b) Plot showing the effect on mission success of an increase in scaling parameter, c, given R_{search} and T_{LOCK}. Changes in R_{search} have a greater effect on the probability of mission success.

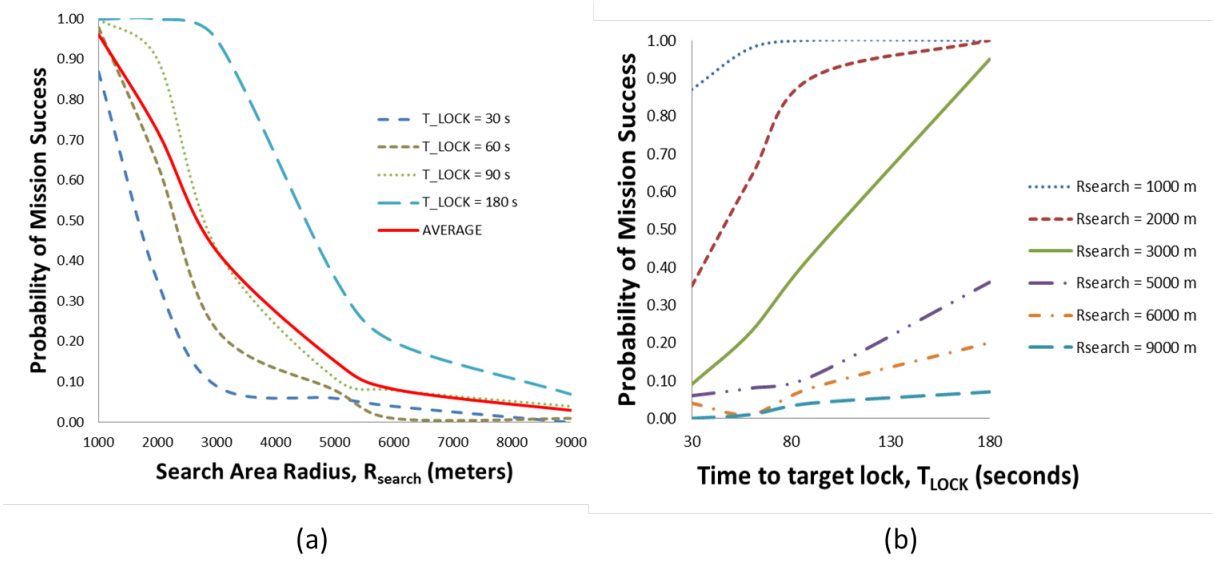


Figure 4.4: (a) Plot of probability of mission success versus search area radius for a given T_{LOCK}. (b) Plot of the probability of mission success versus T_{LOCK} for a given search radius.

Further, the probability of mission success decreases in an exponential manner as the radius of the search area increases. This is partly imposed by the operating conditions of the simulated searcher. By confining the searcher to a finite area, the Levy distribution on leg lengths is

T_{LOCK}	R_{search} [meters]	$\mathbb{E}[T_D]$ [minutes]	$\mathbb{E}[T_C]$ [minutes]	$P(Success)$
30	**1000**	1.7	4.6	**0.87**
30	2000	3.2	4.5	0.35
30	3000	6.0	3.8	0.09
30	9000	N/A	2.8	0.00
60	**1000**	2.4	1.2	**0.98**
60	**2000**	7.5	16.9	**0.64**
60	6000	14.4	6.5	0.01
60	9000	1.3	4.5	0.01
90	**1000**	3.4	0	**1.00**
90	**2000**	14.9	14.3	**0.90**
90	3000	18.4	17.5	0.43
180	**1000**	3.5	N/A	**1.00**
180	**2000**	18.7	N/A	**1.00**
180	**3000**	43.2	56.1	**0.95**
180	5000	35.0	61.1	0.36

Table 4.5: Sample of the results of $n = 100$ simulation runs against an anti-UAV asset with variations on T_{LOCK} and R_{search} for constant Levy distribution parameters. Mission success is achieved upon target detection, with mission failure occurring if the searcher is counter-targeted. Parameters leading to a higher probability of mission success than mission failure are bold-faced.

effectively cut off at twice the area radius, as discussed in Section 2.3.2. As R_{search} is increased for a defined distribution on leg lengths (constant c), more of that distribution is "exposed," increasing the overall possibility that a search leg exceeding l_{crit} is flown.

4.4 Providing Decision Support

Given the insights provided by the numerical studies, we can further develop and easily construct reference diagrams that can provide decision support to commanders assessing the search capability versus asset vulnerability trade off in conducting associated surveillance missions.

The simulation developed throughout this research is useful in that it can quantify measures of searcher performance given a concise set of operating parameters. We conduct further simulations, separating T_D and T_C, in order to determine an analytic means for prediction of searcher performance through regression of the simulation results. The intent is to separately measure searcher performance with respect to target detection time, based on searcher movement and sensor capability, and counter-targeting time, based on adversary capability and searcher movement, then realte their statistics to each other to determine an overall probability of mission

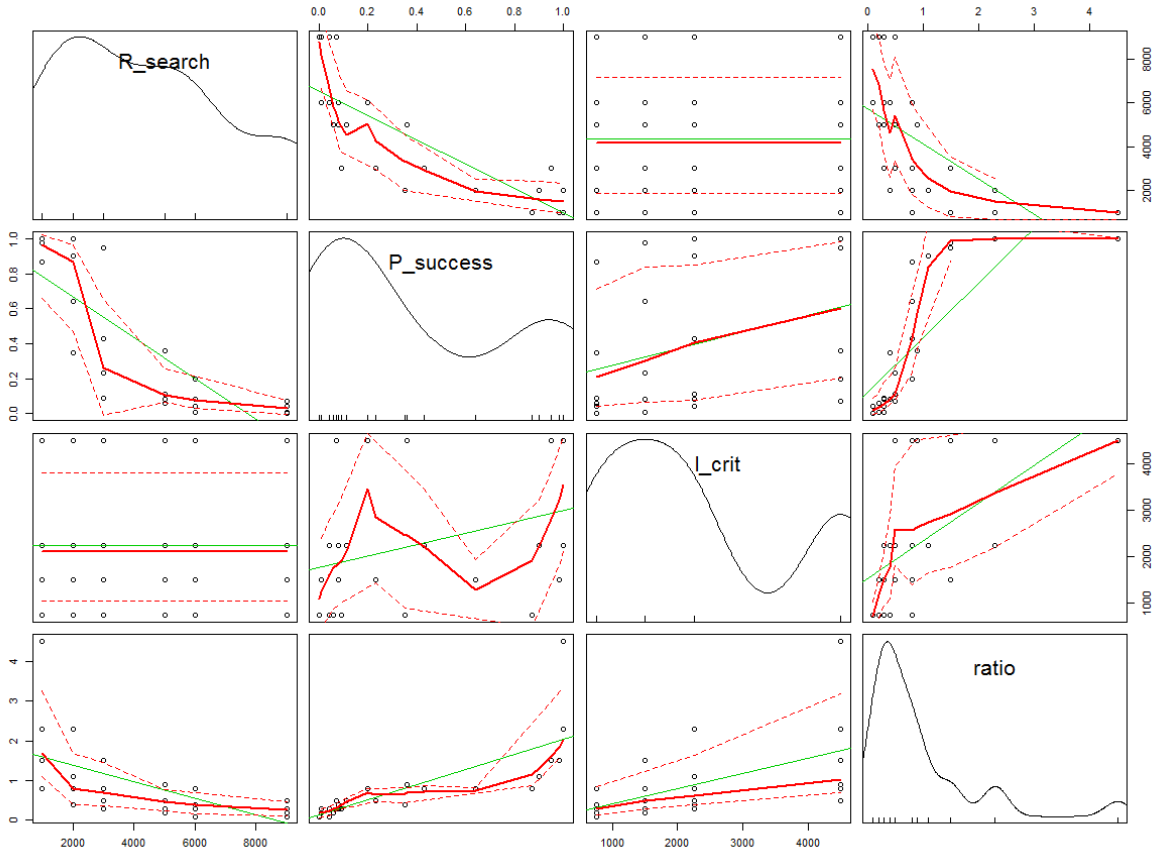


Figure 4.5: A pairs plot of the data is generated in *R* to assist in quantifying the relationship between the search area size, target lock time, and the probability of mission success.

success.

4.4.1 Operational Viability of Nondeterministic Search Patterns

As a preliminary step to first convey tactical relevance of nondeterministic search patterns as proposed by this work, we compare a nominal study of the Levy-based search with the expected time to detection performance for idealized search. To generate a working model of the Levy-based coverage, the simulation is executed for $n = 10,000$ replications. For these studies, the counter-targeting capability of the enemy is disabled in the simulation, such that the search progresses until the stationary target is found. This set of simulations employs the same values for searcher speed and sensor capabilities as shown in Table 4.3, with R_{search} at the 6000-meter level, derived from the maximum area able to be uncovered by the same searcher performing a continuous ideal (i.e., non-overlapping) search. The shift parameter of the Levy distribution, μ,

is again set at four times the turn radiusto facilitate the use of the Dubins path algorithms, and Levy scaling parameter, c, is set to 0.5. A histogram on the times to target detection is presented in Figure 4.6.

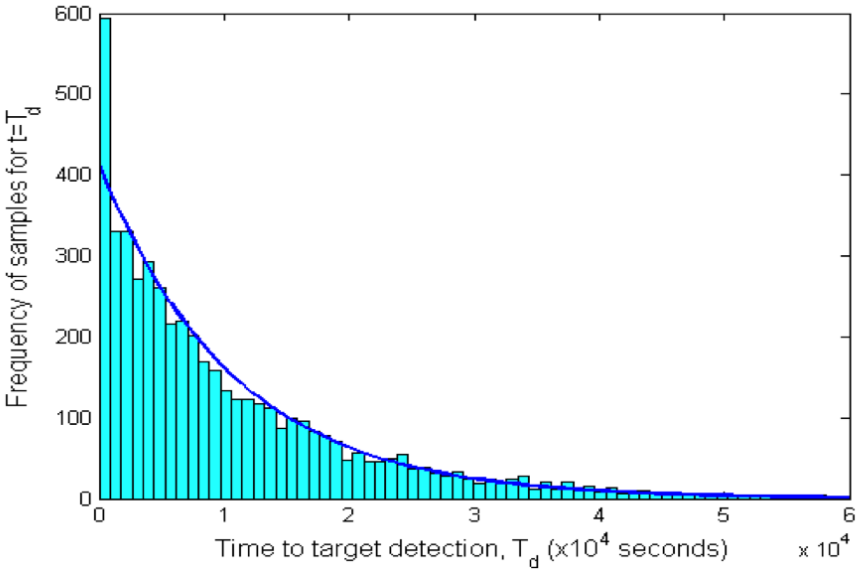


Figure 4.6: Histogram on time to target detection, T_D, resulting from $n = 10,000$ simulation runs without enemy counter-targeting capability. The dark line function represents an exponential regression fit on the distribution.

We conduct a regression analysis on the distribution of these results, applying an exponential fit to model this probability density function on the time to (first) target detection for the given search scenario parameters and noting the assumption of the "cookie-cutter" detection model. The resulting fitted model for the probability distribution is given by:

$$P(T_D = t) = e^{-0.00009095t}. \tag{4.3}$$

This regression fit is represented by the dark line in Figure 4.6. Having constructed a metamodel for the probability distribution function for this set of searcher parameters, we can compute the

estimated time to target detection, $\mathbb{E}[T_D]$.

$$\begin{aligned}\mathbb{E}[T_D] &= \int_{-\infty}^{\infty} t f(t) dt \\ &= \int_{0}^{\infty} t\ e^{-0.00009095t}\ dt \\ &= 3.1 \text{ hours.}\end{aligned} \tag{4.4}$$

On average, utilizing the parameters established for this set of simulations and ignoring any counter-targeting capability of the enemy, we expect the target to be detected in 3.1 hours. For comparison, we calculate T_D for a deterministic "spiral-out" search pattern illustrated in Figure 4.7. This pattern is a variation of the perfect "lawnmower" search pattern described in Section 2.1, with the searcher beginning at the center of the search area and conducting straight-line search legs marked by 90° turns, moving outwards from the origin until the entire search area is cleared. The spiraling searcher employed in this comparison is assigned the same sensor

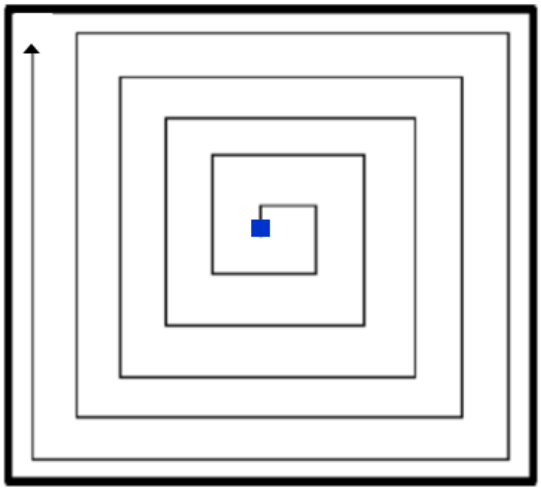


Figure 4.7: A variation of the "lawnmower" search discussed in Section 2.1, the "spiral-out" search begins at the origin and conducts straight-leg, *predictable*, non-overlapping sweeps toward the search boundary edge.

and speed characteristics as the searcher in the Levy-based search example. In this systematic search pattern, the time to target detection can be explicitly computed (given target location) using the searcher speed and the sum of the straight-line distances traveled before the target is within range of the sensor. Table 4.6 shows the deterministic time to detection for targets placed at various distances from the origin.

Target distance from origin [meters]	Ideal T_D [hours]
1000	0.2
2000	0.6
3000	1.1
4000	1.9
5000	2.9
6000	4.1

Table 4.6: Times to target detection utilizing a deterministic "spiral-out" search, dependent on distance of the target from the beginning of the search.

The $\mathbb{E}[T_D]$ of 3.1 hours calculated from the simulation results is within an acceptable magnitude for comparison to this deterministic method of search. These results indicate that the detection performance of the Levy-based search pattern is comparable to that of the continuous search pattern.

4.4.2 Estimating time to counter-targeting, T_C

To now provide guidance or "rules of thumb" based on the impact of counter-targeting, we can revisit the Levy-based search simulation with varying values of T_{LOCK}. The same searcher parameters as previously used (displayed in Table 4.7), are applied to $n = 1000$ replications of the simulation. To specifically investigate T_C, this set of simulations is conducted with no target present in the search area, such that the search progresses until a straight-line leg length exceeding l_{crit} is flown, resulting in the searcher being counter-targeted.

Variable	**Value**	**Units**
T_{LOCK}	$\{30, 60, 90, 180\}$	[seconds]
v	25	[meters/sec]
c	0.5	[none]
μ	200	[meters]
R_{search}	6000	[meters]
r_{sensor}	161	[meters]

Table 4.7: Input parameters for simulation runs to assess T_D and T_C.

Figure 4.8 shows the cumulative probability of the searcher being counter-targeted for each simulated value of T_{LOCK}, scaled to the upper 20% of the probability density to better display the results. This graph represents the probability that counter-targeting of the searcher occurs at

or before time t for a given T_{LOCK}. We can use this cumulative distribution graph to determine a probability of mission success.

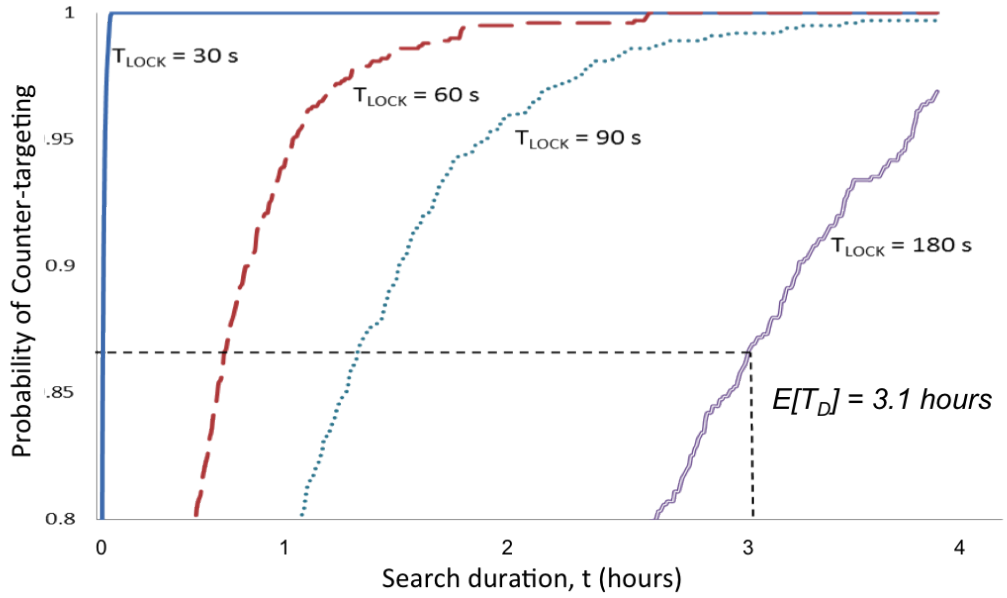


Figure 4.8: This summary plot of the cumulative distribution functions for varying values of target lock time ($n = 1000$) can be used to determine a measure of the probability of mission success.

As an example, we plot the expected time to target detection, $\mathbb{E}[T_D]$ calculated in Equation 4.4, against the cumulative probability of counter-targeting versus an anti-UAV asset with a $T_{LOCK} =$ 180 seconds. We can read off a surrogate measure of the probability of mission success (c.f. Equation 4.1) by using the *expected* time to detection, which is known *a priori* by the mission commander, since the search parameters are nominally dictated by the search platform:

$$P(\mathbb{E}[T_D] < T_C) = 1 - P(T_C \geq \mathbb{E}[T_D])$$

Using $\mathbb{E}[T_D] = 3.1$ hours, we can estimate, based on the expected value graphs of the probability of counter-targeting (Figure 4.8), that the probability of mission success is $1 - 0.86 = 0.14$ or 14%.

Referring back to Table 4.5, for a T_{LOCK} of 180 seconds, the expected probability of mission success in a search area of 6000 meter radius is 20%. As expected, the two methods to determine this measure of performance produce fairly comparable results, and as such, provide the mission commander a means to quantify the probability of mission success for a given set of searcher and search area parameters.

4.5 Field Experimentation

In this thesis research, we leveraged unique opportunities to conduct real-world flight testing of the Levy-based search algorithm in live-fly contexts. As part of the SECNAV's initiative for the Consortium for Robotics and Unmanned Systems Education and Research (CRUSER) [47] and the Naval Postgraduate School's Advanced Robotic Systems Engineering Laboratory (ARSENL), live-fly field experimentation was conducted at McMillian Airfield, Camp Roberts, CA. The demonstration of the search pattern was conducted as part of a multi-UAV search experiment during the 13-2 Joint Inter-agency Field Experimentation (JIFX) events, an effort by the Naval Postgraduate School to explore capability gap solutions and new technologies in support of the operational needs homeland defense through discovery, exploration, and experimentation.

We implement functionality in **LookAheadLevyFlight.m** to store the major Levy waypoints traversed during the simulated search in Cartesian coordinate form. These "x" and "y" coordinates are input to **LevytoLatLon.m**, along with a desired starting latitude and longitude representing the center of the search area. **LevytoLatLon.m** iterates the Levy waypoint array and converts the distance between each Cartesian waypoint to a Great Circle distance, returning the coinciding latitude and longitude points in an ".fpf" file format for uploading into the Procerus *Virtual Cockpit* (VC) ground control station software. The waypoints in VC are transmitted to the Procerus Unicorn UAV employed during these flight experiments, shown in Figure 4.9.

Figure 4.10 shows the straight-line path between the major Levy waypoints generated during one run of the simulation model. McMillan Airfield is shown to the east of the established search area bound within the white dashed circle. The input parameters to each of the four UAV flights are listed in Table 4.8. The altitudes of the UAVs were staggered from 300 to 600 meters to ensure altitude deconfliction for safety of flight.

Variable	**Value**	**Units**
R_{search}	1200	[meters]
c	0.5	[none]
μ	200	[meters]
v	15	[m/s]

Table 4.8: Searcher parameters established for real-world flight during multi-UAV JIFX demonstration.

Figure 4.11 shows the actual flight path taken from the searcher. Appendix C details the procedure for recovering telemetry data from the Procerus software and converting to a .kml file

Figure 4.9: Procerus Unicorn UAV [48] used in multi-UAV *look-ahead* Levy search demonstration conducted at 13-2 JIFX in February, 2013.

for display in Google Earth. The UAV executes flight trajectories with bounded curvature, consistent with the Dubins curves generated by our simulation, as shown in the magnified flight segment displayed in Figure 4.12. However, forcing the UAV to a Dubins flight path requires an excessive number of waypoints for any significant duration of flight. The overlay of the actual flight trajectory compared to the Dubins path generated by **LookAheadLevyFlight.m** is sufficient evidence that our simulation adequately models the flight physics of a small UAV.

During the demonstration, ARSENL members were able to employ four UAVs simultaneously conducting the *look-ahead* Levy search. The flight trajectories are illustrated and distinguished by color in Figure 4.13. This experiment represents significant accomplishments in implementation and system integration, and also highlights the apparent advantage of employing multiple searchers evidenced by substantial coverage of the search area.

4.6 Bayesian Update and Looping Function

Based on the significant simulation and field experiment studies, key avenues for improving the performance of search as well as its operational relevance include the application of Bayesian methods to incorporate imperfect sensing and prior intelligence models for likely locations of the stationary target. Further, the use of adaptive search, where decisions on where to search are

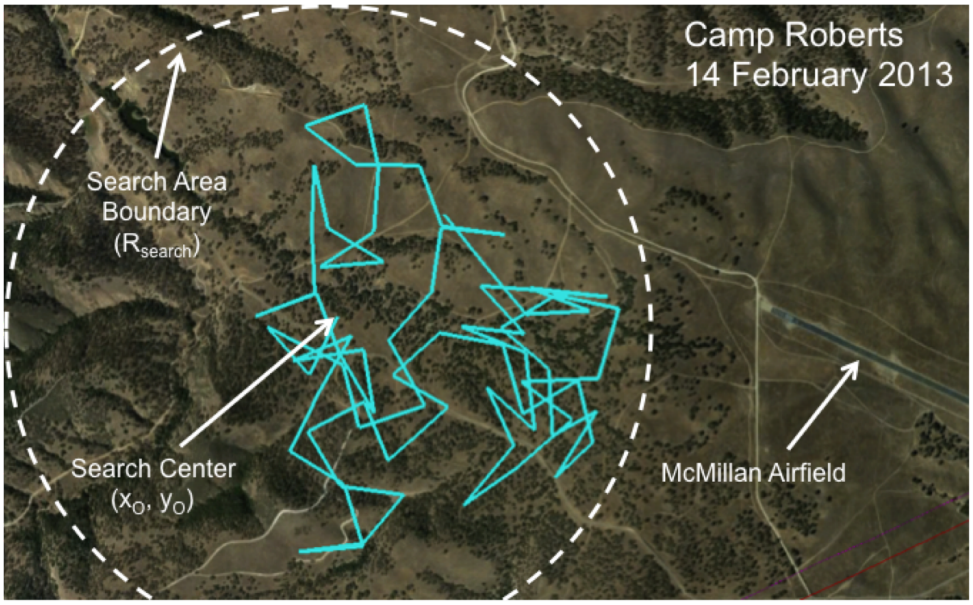


Figure 4.10: Google Earth overlay of major Levy waypoints generated by **LookAheadLevyFlight.m** algorithm. The starting airfield, search center, and search boundary are annotated.

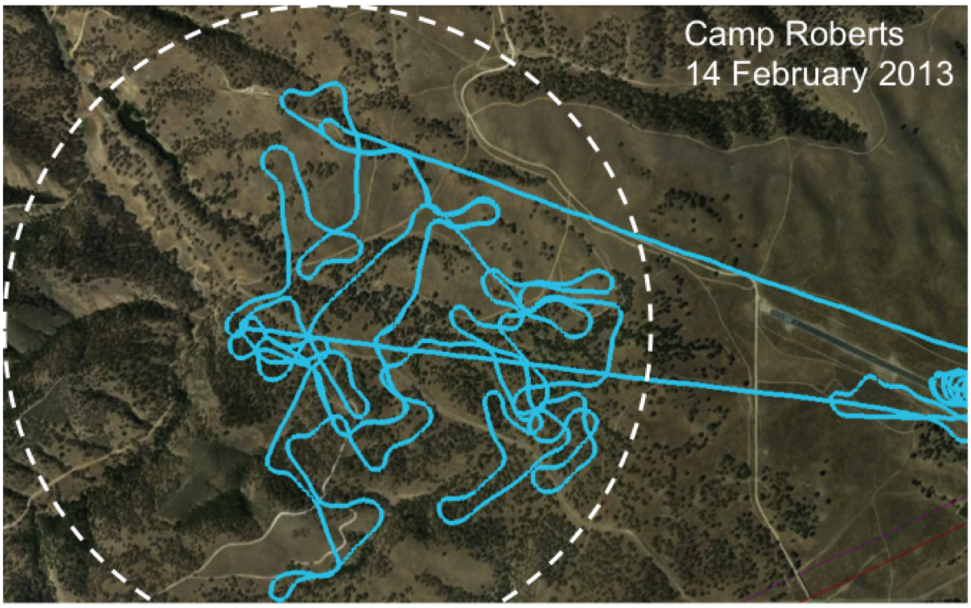


Figure 4.11: Google Earth overlay of the actual flight path of a single UAV searcher conducting the *look-ahead* Levy search.

determined based on the results of previous search effort, is known to be significantly advantageous in search [9, 49]. We explore such enhancements to the proposed approach, enabling preliminary assessment of their value to the nondeterministic search process.

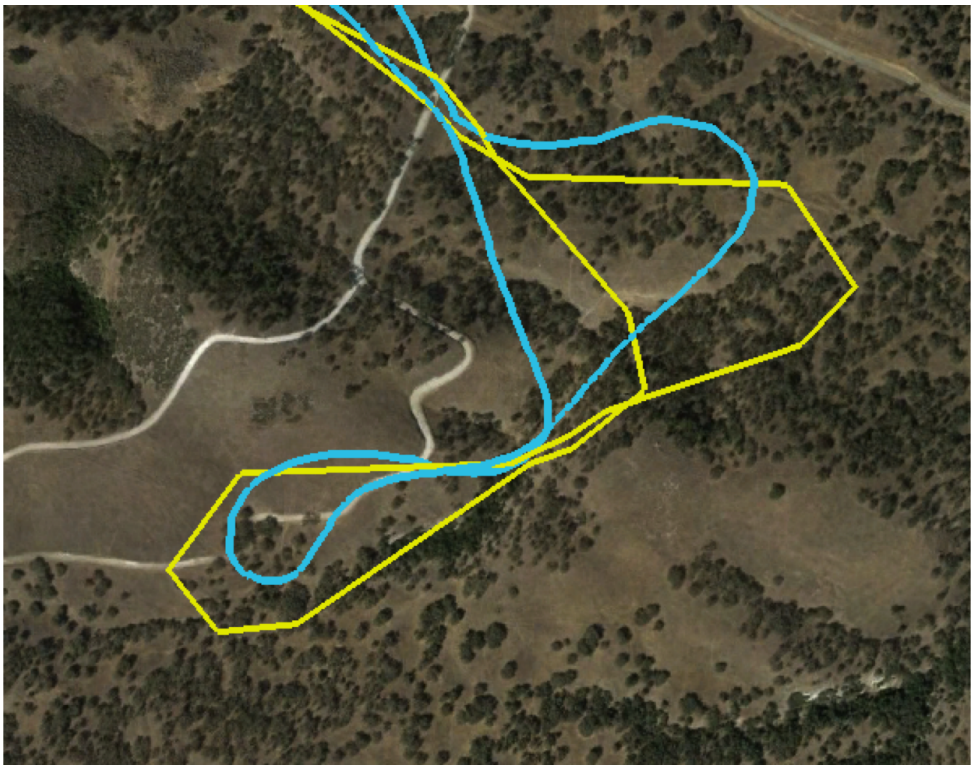


Figure 4.12: A magnified section of the flight path traveled by the UAV compared to the waypoints generated through the use of Dubins curves in **LookAheadLevy.m**. The simulation model created for this thesis is demonstrably an adequate representation of the flight physics of a small UAV.

A Bayesian update scheme is applied to the simulation model, with a "looping" ability which imparts a bias on the searcher direction based on iteratively updated probabilities of target location, thereby potentially decreasing the time to target detection. As described in Section 2.2.2, this Bayesian update process utilizes the prior probability that a target exists in the given cell and updates that probability based on whether a detection occurs when the searcher "looks" in that cell. This Bayesian update process allows for the incorporation and analysis of the characteristics of an imperfect sensor, i.e., the probability of false alarm, α, and probability of missed detection, β. We apply the Bayesian update scheme (where probability mass is shifted in accordance to observations), though we make use of a perfect sensor in order to gain insight into the exploitation of the biasing (i.e., looping) that results from the Bayesian update, as discussed in detail in Section 2.2.2. For better visual representation and decreased simulation run time, we scale down the search parameters to those shown in Table 4.9.

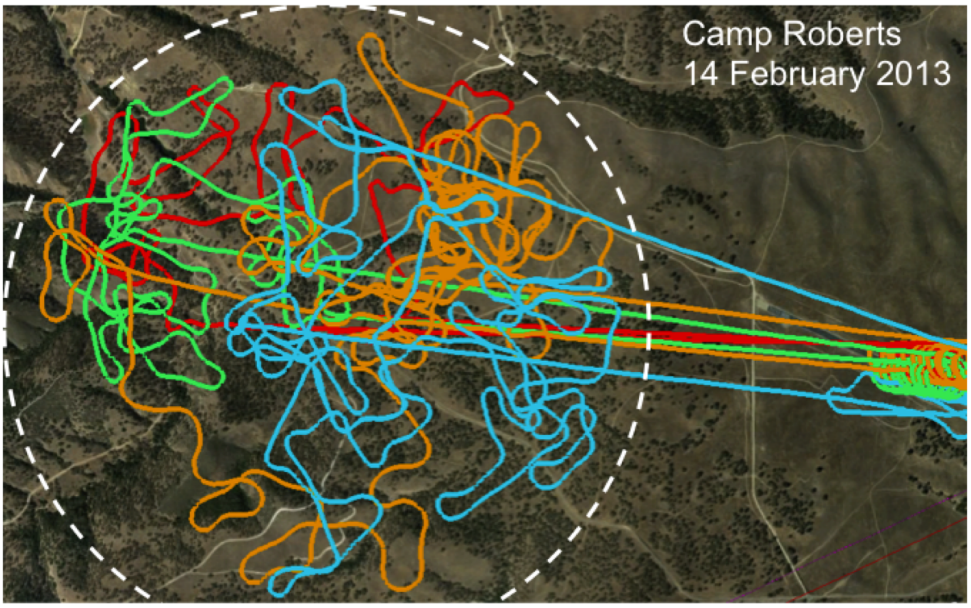


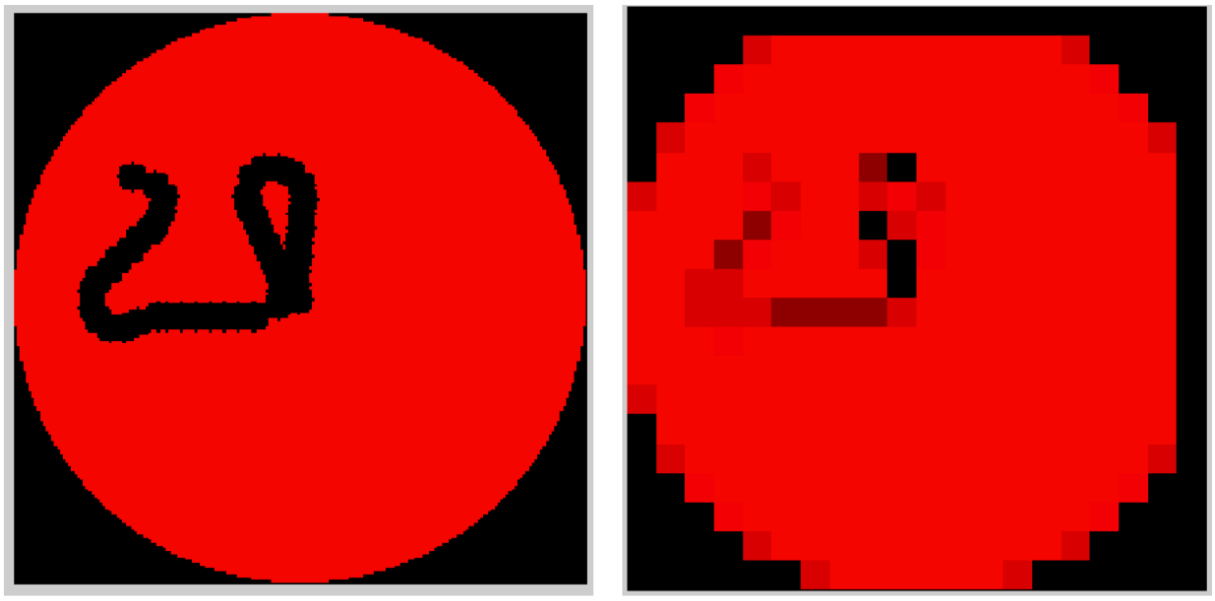
Figure 4.13: Google Earth overlay of the actual flight path of a single UAV searcher conducting the *look-ahead* Levy search.

Variable	**Range**	**Units**
T_{LOCK}	90	[seconds]
r_{sensor}	54	[meters]
r_{turn}	25	[meters]
μ	100	[meters]
c	0.5	[none]
R_{search}	1500	[meters]
α	0.0	[none]
β	0.0	[none]
t_{loop}	300	[seconds]

Table 4.9: Inputs to the simulation incorporating a Bayesian update and looping function.

As shown in Figure 4.14, the "binning" function of the **LevyLoop.m** algorithm tracks the probability density for discrete areas of the search grid. This running calculation is used to send the searcher to clusters of high probability density upon expiration of the user determined loop timer. In these simulation runs we employ a five minute loop timer.

To compare the performance of the Bayesian looped look-ahead search to that of the standard



(a) (b)

Figure 4.14: (a) Heat map showing the distribution of probability of target presence employing a perfect sensor Bayesian search over the area. (b) The probability density of larger search area sections is used to bias the movement of the searcher upon expiration of a loop timer.

look-ahead Levy search used for the majority of this research, we randomly draw a target position (x_{target}, y_{target}), and run the simulation multiple times for each algorithm. The results of average detection time and mission success are shown in Table 4.10.

	P(success)	$T_{D,avg}$	$F_{avg}(T_D)$
Standard Look-ahead Levy	1.00	1221	0.59
Looping Look-ahead Levy	0.90	480	0.35

Table 4.10: Comparison of performance between the standard Look-ahead Levy search and a *look-ahead* Levy search implementing the looping function and Bayesian update. The looping function reduces the average time to target detection, but the forced override of the length distribution sometimes leads to exceeding the critical length, resulting in counter-targeting of the searcher, and overall mission failure.

The looping function overrides the random Levy length and heading angle distributions and directly and manually steers the searcher toward the area of highest probability density. As shown by the slightly lower probability of success, this looping bias may cause the searcher to exceed the critical length and thus be counter-targeted. However, the average time to target detection when implementing the looping function is almost one-third of the time of the standard

look-ahead Levy search. As an added benefit of the looping function, on average, only half as much of the search area needed to be uncovered to reveal the target location. This improvement may have implications on counter-targeting probability itself, due to less "fly-over" of contested territory. These results reiterate the need for accurate intelligence regarding enemy capability. By including the looping function into the *look-ahead* Levy search, but limiting the maximum straight-line distance flown by the searcher during the biased relocation to l_{crit}, the possibility exists to reduce target detection time and exposure to adversarial assets while maximizing the probability of mission success. Such assessment and analysis is left for follow-on studies that can leverage these benefits.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Future Avenues

5.1 Conclusion

The work accomplished in this thesis provides insight as to the utilization of nondeterministic search patterns to counter anti-access area denial (A2AD) tactics becoming more prevalent in foreign operations. The Levy distribution on leg lengths serves as only one particular avenue to achieving a successful search over this contested territory. The benefits against counter-targeting of the searcher afforded by the Levy distribution reside in the ability of the user to adjust the Levy scaling parameter, c. Recall from Section 1.3 that as the value of c is increased, the uniformity of leg lengths returned from the Levy distribution increases, resulting in a longer average search leg length. For these higher values of c, the searcher has a greater tendency to traverse the entirety of the search area during a span of limited endurance, likely resulting in higher percentage of area covered, $F(t)$, for a given search time. However, traveling these longer leg lengths results in an increased probability of being counter-targeted due to exceeding l_{crit}. For the sizes of search area explored in this research, changes in c provided only small differences in the probability of mission success, albeit with a linear trend indicating that large c over a large search area will result in a higher probability of mission failure. A small c value ($c \leq 1$) provides the highest probability of mission success operating under the parameters simulated in this research, since this ensures that the majority of Levy probability density is near the μ value, allowing for the implementation of Dubins curves and retaining short flight legs, thereby minimizing the probability of exceeding the critical length of the target lock time.

The Levy distribution has been established as the optimal search pattern when locating mobile targets over an infinitely bound area [18, 26]. However, the limitations imposed on the searcher's travel by the finite area boundary used for this thesis may impact the optimality of the Levy search, as shown by little difference in performance for changes in Levy parameters. Future works focusing on high endurance searchers capable of searching long ranges must be undertaken prior to labeling the Levy as the optimal distribution on leg lengths for the scenario set here in. Thus, the main contribution of this thesis is the development of the simulation model capable of examining various distributions on search leg length and searcher heading. The scenario specific to this research focused on an airborne search of a flatland area. However, this model has been developed with a robust searcher parameterization capability, and can be

applicable to a variety of searcher platforms, from airborne to undersea.

5.2 Avenues for Future Work

The development of a simulation capable of analyzing performance for various nondeterministic search pattern distributions encompassed the majority of the duration allotted for this research. As such, there exists several avenues of future work available for continuation of the efforts described in this thesis. An important aspect of a search simulation is the incorporation of realism into the model, both operationally and environmentally. There are several extensions that can be added to this simulation model to improve upon that realism.

From an operational perspective, the incorporation of multiple searchers and multiple targets, both moving and stationary, will add increased realism and operational relativity. The Levy distribution has been shown to outperform the "lawnmower" style continuous search, as well as Brownian searches, when attempting to locate multiple moving targets using multiple searchers [23, 24, 34, 50, 51]. The field experiments conducted for this thesis (Figure 4.13) demonstrate that the simultaneous employment of multiple searchers traveling on nondeterministic search paths will result in a higher area coverage over a finite time period. This was simply an example of multiple non-cooperative searches traveling paths derived from the same algorithm. Introduction of cooperative functions between these searches could serve to further increase the benefits of the nondeterministic search [52]. The incorporation of these cooperative searchers can lead to further investigation on proper searcher allocation once a target is found, especially when it is expected that multiple targets are within the search area [53, 54]. The incorporation and analysis of various distributions on searcher leg length and heading angle will provide insight as to the optimal nondeterministic pattern when operating in a finite search area.

Modeling the shape and size of the searcher and targets would greatly effect the product of the simulation. Accounting for radar cross section of the searcher relative to a known anti-UAV asset can increase fidelity on the measure of counter-targeting time where as increasing the size of the target beyond the simple meter squared metric utilized in this research may greatly improve time to target detection.

The enhancement of the operating environment itself will add greater realism to the simulated scenario. There have been several studies suggesting algorithmic solutions to obstacle avoidance of unmanned searchers [52, 55, 56]. Incorporation of physical geography and integration of these avoidance algorithms would reveal limitations on UAV flight, and would be particu-

larly useful when translating this algorithm into use for a ground searcher. Work accomplished regarding optimal coverage using a Dubins vehicle could be extended and incorporated into this research by changing the shape of the search area to account for detailed territorial borders or geographic constraints establishment, and analyzing the area coverage when a smooth circular border is not established [57].

Further, adding weather factors such as course drift due to wind or sensor degradation due to visibility limitations would also increase the realism of the simulation.

5.3 Operational Impact and Recommendations

Since the beginning of the new millennium, the United States has steadily increased reliance on unmanned vehicles for conduct of critical missions. These platforms provide a means for infiltrating contested territory, for conduct of strike or surveillance operations, without directly endangering personnel. However, the world has taken notice of this trend in utilization of unmanned searchers and, as a result, our adversaries have begin focusing training and technology on anti-UAV programs, employing anti-access area denial strategies determined to thwart U.S. efforts in intelligence, surveillance, and reconnaissance operations. Perhaps even more significant, is the subdued sense of conflict associated with destroying these inanimate objects, with capturing of U.S. drones being used recently as a bolster of pride, where as the downing and capture of a U.S. pilot would most surely be considered an act of war that many countries are not willing to instigate. It is imperative to the longevity of these unmanned programs that we develop tactics aimed at mitigating the A2AD tactics of our adversaries. The consumers of the services provided by these UAVs and other unmanned platforms need to take a step back and look toward the future, at the increasing threats and how we can best deploy stealth and randomness to mitigate them. The insights contrived from this thesis can help lead the discussion as to strategies and technologies we must develop in order to maintain the technical and operational edge over our adversaries, not only in the development of further unmanned systems, but in our own research and engineering of anti-UAV weapons.

At the tactical level, this thesis develops a simulation model capable of informing the mission commander regarding employment of nondeterministic searchers in a contested territory, an increasingly common mission in today's operational environment. The production of decision support tools aimed at shaping the answers to questions such as what specific platform is suited for a particular search mission, or how many of that platform need be deployed to provide optimal results. Perhaps most importantly is the commanders ability to place a quantifiable

estimate on the probability of achieving a mission success given a baseline intelligence report regarding enemy capabilities.

APPENDIX A:
Summary of Variables and Symbols

The MATLAB simulation developed for this thesis incorporates a robust parameterization capability, allowing for the exploration and analysis of multiple searcher types, search leg/angle distributions, and search area characteristics. The simulation inputs which can be set by the user are listed in Table A.1. The variables used as outputs to indicate search performance are shown in Table A.2.

Table A.1: Simulation variables set by the user

Variable	**Description**
Display Options	
`plot_ on`	set to '1' if any graphical display is desired
`flight_ path_ on`	set to '1' if flight path track is to be displayed
`l_ crit_ warning_ on`	set to '1' if warning meter for exceeding critical length is to be displayed
`coverage_ plot_ on`	set to '1' if area coverage plot as a function of time is to be displayed
`plot_ 3D`	set to '1' if 3D plot showing relative altitude is desired
Output Options	
`generate_ Lat_ Lon`	set to '1' if Cartesian waypoints generated during simulation are to be converted to decimal lat/lon and loaded in `.fpf` file for input to Procerus Virtual Cockpit
`filename_ Levy`	input filename for general look-ahead Levy waypoints - default is '`LevyWaypoints.fpf`'
`filename_ Dubin`	input filename for waypoints forced to adhere to Dubins path - default is '`DubinsWaypoints.fpf`'
Functional Options	
`loop_ levy`	set to '1' to implement looping function, sending searcher to highest probability density area after defined `t_ loop` time

Continued on next page

Table A.1 – continued from previous page	
Variable	**Description**
t_ loop	set time associated with looping function (how long to search before moving to area of highest probability density)
iteration_ count	set the number of simulation runs to conduct for the current set of parameters
coverage_ on	set to '1' if coverage calculations are to be completed (setting to '0' turns off coverage if desired to save run time and focus purely on detection or counter-targeting parameters)
target_ detect_ on	allows user to turn off the target detection piece in order to focus on coverage and/or counter-targeting
bayesian_ on	set to '1' if Bayesian calculations are to be conducted (setting to '0' assumes a perfect sensor and saves time/memory on detection calculations)
counter_ target_ on	allows user to turn off the target detection piece in order to focus on coverage and/or target detection
Movement Characteristics of Searcher	
v	speed of searcher in meters per second
altitude	altitude of the searcher corresponds to 2000 ft for UAV in meters
r_ turn	minimum turn radius of the searcher in meters
UAV_ endurance	maximum endurance of the searcher in meters
c	scaling parameter, determines the peak distribution of the Levy curve, hence the lengths of each search leg
Sensor Characteristics of Searcher	
fov	field of view of the sensor, in conjunction with altitude, determines sensor footprint. Default is 30 degrees as employed by a typical small UAV
a	sensor probability of false detection
b	sensor probability of missed detection
p_ confirm	grid cell probability above which a target is confirmed to be located in a given cell, c
Continued on next page	

Table A.1 – continued from previous page	
Variable	**Description**
p_ deny	grid cell probability below which a target is confirmed not to be located in a given cell, c
Counter-targeting Parameters	
target_ lock	intelligence-based time required for the anti-UAV asset to gain target lock and destroy UAV searcher
Search Area Parameters	
R_ search	radius of the circular search area in meters
delta_ dist	defines the distance between "looks" in meters. In a perfect simulation this number would be 0, as the search would be continuous. In order to expedite the running of the program we use a default step distance of r, returning a minimum 96% coverage of the area flown over during each search leg

Table A.2: Variables which are populated throughout the simulation with information useful to the analysis of search performance

Variable	**Description**
Coverage Variables	
`coverage_plot`	Matrix representing discretized coverage area. Each cell will be '0' if not searched, '1' if searched
`coverage`	Matrix to store percent of area covered at the end of each search leg, can store multiple iterations of simulation
`prob_ density`	Matrix representing discretized coverage area. Each cell will hold probability of target being located in that cell
`time`	Matrix which stores the time stamp at the end of each search leg; Indices map directly to coverage in 'coverage; matrix
Performance Metrics	
`T_d_array`	Array to store final time to detect the target ('0' if no detection occurs)
`T_c_array`	Array to store final time for searcher to be counter-targeted ('0' if no counter-targeting occurs)
Nondeterministic Search Parameters	
`angles`	Array to store angles traveled as a result of heading changes drawn from the uniform distribution
`distances`	Array to store Levy lengths applied to each search leg
`DubinsWaypoints`	Matrix to store all x-coordinates, y-coordinates, and altitude of intermediate waypoints along Dubins path
`LevyWaypoints`	Matrix to store all x-coordinates, y-coordinates, and altitude of major Levy waypoints only

APPENDIX B:
Software and Hardware

The following computing software and hardware was used in the development of the search model and analysis of simulation results:

Software:

- Excel®
  Microsoft®Office®Professional Plus 2010

- RStudio Statistical Software
  Version 0.95.258

- Mathworks®MATLAB®R2012a
  Version 7.14.0.739

  The following MATLAB code was generated during development of this simulation. The description of each function is discussed in detail in Chapter 3. Each can be made available upon request at `http://faculty.nps.edu/thchung/`
  - LookAheadLevyFlight.m
  - LevyGenerator.m
  - LevyLength.m
  - Dubins.m
  - DubinsLSR.m
  - DubinsRSL.m
  - DubinsTurnAround.m
  - LevyDetection.m
  - LevyCoverage.m
  - LevyLoop.m
  - LevytoLatLon.m
  - circle.m
  - spasf.m

Hardware:

- ARSENL-Lab6
  Intel®Core™i7-2700K CPU @ 3.5 GHZ
  32.0 GB

- Mac-C1MHH11PDV13
  Intel®Core™i5 CPU @ 2.4 GHZ
  4.0 GB

- PC-DELLXPS8300
  Intel®Core™i7-2600K CPU @ 3.4 GHZ
  12.0 GB

The following software was used in the conversion of simulation data to real world GPS data for the conduct of field experimentation and the display of the resulting telemetry data: Software:

- Procerus Technologies®Virtual Cockpit™
  Version 2.6.0 build 3 protocol 8.0.1

- Google®Earth©
  Version 6.2.2.6613

- GPSVisualizer.com®
  `http://www.gpsvisualizer.com`

Hardware:

- Procerus Unicorn UAV

APPENDIX C:
Waypoint Data Processing for Field Experiments

This appendix summarizes the procedures for generating waypoint data for use in the Camp Roberts field experiments, as well as the conversion of telemetry data collected from those experiments to .kml files for display in Google Earth.

C.1 Waypoint Generation

The generation of a waypoint file in .fpf format, for upload directly to Procerus *Virtual Cockpit*, is a functionality built directly into the **LookAheadLevyFlight.m** algorithm. In the section of the code commented as "output options" there is a Boolean variable named *generate_ Lat _ Lon*. Setting this variable to '1' activates two print scripts, one which writes the main Levy waypoints to an .fpf file, the other that writes every waypoint used in the generation of the Dubins path flight to a separate .fpf file. It is important to note when attempting to upload these waypoints to Virtual Cockpit that the Dubins file can contain in excess of ten times the number of waypoints as the Levy waypoint file.

The names of the output files can be defined by the user by changing the value of the *filename_ Levy* and *filename_ Dubin* string variables in this same section of the **LookAheadLevyFlight.m** code. The default names of the files are "LevyWaypoints.fpf" and "DubinsWaypoints.fpf". Run the **LookAheadLevyFlight.m** to create and populate theses files, which will be output to the main MATLAB directory.

To upload the waypoints, in Virtual Cockpit click 'File', 'Load Flight Plan', then select the generated .fpf file from the MATLAB directory. Once uploaded, clicking the up arrow at the bottom center of the Virtual Cockpit display screen will reveal the list of waypoints, showing speed, altitude, latitude, and longitude. Any necessary adjustments can be made here prior to transmitting to UAV.

C.2 Display of Waypoint Data

The format of the .fpf data file input to Virtual Cockpit requires that each line of waypoint information be followed by a blank line, effectively double spacing the waypoints. Unfortunately, `gpsvisualizer.com` cannot recognize continuity between the waypoints if the line is skipped, and the resultant .kml file will plot each waypoint, but will not connect a track between them.

To prevent this, we developed a macro for Excel which simply removes the blank lines between waypoints, as well as the extraneous header data required by Virtual Cockpit, but not suitable for display in a .kml file. To compress the .csv waypoint file for transformation to a .kml file, open the .fpf file generated by **LookAheadLevyFlight.m** in a text editor. Copy all the waypoint information into a .csv file in Excel, then run the macro.

Once the UAV has conducted the Levy flight and the telemetry data has been downloaded, the data can be sorted and parsed for use in the generation of a .kml file, able to be displayed as waypoint tracks in Google Earth. The telemetry information returned from the UAV contains 108 columns of data, with one row assigned for each telemetry reading (thousands of rows). To generate the .kml file, only the basic flight data is require - Heading, Speed, Altitude, Latitude, and Longitude. The following steps describe the .kml generation process.

1. Open the telemetry file in MATLAB. The file is named with the UAV identification and date, e.g. **telemetry_ 2460 _ 2013_ 02_ 14_ 07.m** is telemetry data for UAV 2460 flown on February 14, 2013. Run this file to initialize and populate the *telemetry.data* variable.
2. Use the 'dlmwrite' command in MATLAB to transfer the telemetry data to an Excel .csv file. Ensure the numerical precision is set to '8' to so that the full latitude and longitude values are transferred.

   $$dlmwrite(\text{'filename.csv'},telemetry.data,\text{'delimiter'},\text{','},\text{'precision'},8);$$

3. Open the .csv file in Excel and run the first parsing macro by clicking the 'Developer' tab, 'Macros', selecting 'TELEMETRY_ PARSE', and clicking 'Run'. This macro combines all time stamp data into and labels each of the columns of telemetry data, placing all of the data in a legible, organized format. This will allow the analyst to choose which data is relevant to their needs. NOTE: This macro must be run to clean up the telemetry data from MATLAB prior to running the .kml generation macro, 'PARSE_ FOR_ KML_ GENERATION'. The VBA code for this macro can be made available from `https://faculty.nps.edu/thchung`.
4. To further parse the data down to only those elements required for .kml generation, run the .kml parsing function by again clicking the 'Developer' tab, 'Macros', and selecting the 'PARSE_ FOR_ KML_ GENERATION' macro and clicking 'Run'. This macro is shown in Listing C.1 and can be made available from `https://faculty.nps.edu/thchung`. This parses the 108 data columns down to the 5 needed to display the flight path in Google

Earth.

5. The easiest way to convert the .csv data to a .kml file is to use an external .kml generator service, such as provided by `http://GPSvisualizer.com`. This website provides free .kml conversion, and offers several options on display, ranging from color and shape of waypoints to 3D altitude representation. Detailed instructions for use can be found on the website.
6. Once the file is generated, double click the .kml file and the track appears in Google Earth.

Listing C.1: Parses telemetry data into compressed .csv for input to gpsvisualizer.com

```
Sub PARSE_FOR_KML_GENERATION()
'
' Parses telemetry data for .kml generation for use in Google Earth
'
    Range("A:AG,AM:DB").Select
    Selection.Delete Shift:=xlToLeft
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Speed"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Altitude"
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "Heading"
    Range("D1").Select
    ActiveCell.FormulaR1C1 = "Latitude"
    Range("E1").Select
    ActiveCell.FormulaR1C1 = "Longitude"
    Range("E1").Select
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D:
Calculations for Discrete Sample Spacing

To save runtime and processing power during simulation runs, we employ a discrete update scheme to the searcher movement. Figure D.2 and the subsequent equations provide the mathematical background to these discrete updates as introduced in Section 3.4.1.

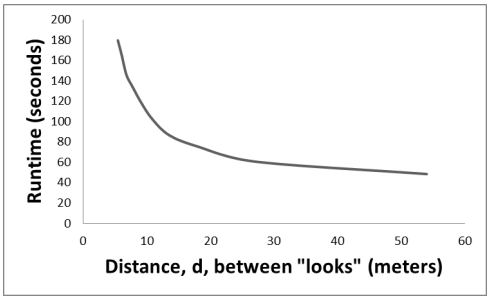


Figure D.1: This graph shows the effects of spacing between the discretized looks on the runtime of the simulation. As the spacing increases, the simulation runtime decreases exponentially. Choosing a discretized spacing equal to the search radius provides 96% of the coverage attained through a continuous search for a fraction of the runtime cost.

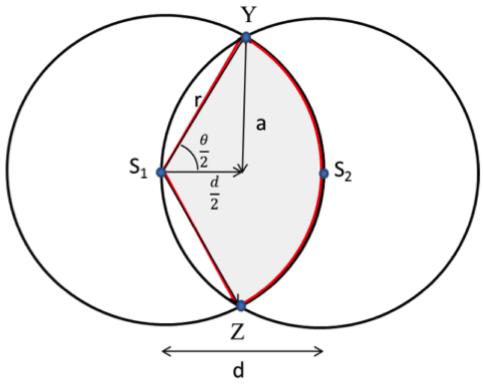


Figure D.2: This illustration shows the geometry involved in calculating the sensor overlap observed during a discretized search with spacing less than twice the sensor radius.

$$A_{lens} = 2A_{wedge} - 2A_{\triangle YZS_1} \tag{D.1}$$

where A_{lens} represents the overlapping portion of the sensor footprint between two subsequent positions of the searcher. A_{wedge} represents the wedge shaped portion of the overlap outlined in

red in Figure D.2 and $A_{\triangle YZS_1}$ represents the triangular portion of the wedge created by points YZS_1. We subtract A_{lens} one time for every two waypoints on the search leg to achieve the total new area covered by the searcher on that leg. The remainder of this section focuses on the derivation of the equations to conduct these area calculations.

The area of the wedge is calculated as follows.

$$\theta = 2\cos^{-1}\left(\frac{d}{2r_{sensor}}\right)$$
$$A_{wedge} = \int_0^{\theta}\int_0^{r_{sensor}} r_{sensor}dr_{sensor}d\theta = \int_0^{\theta}\frac{r_{sensor}^2}{2}d\theta = \frac{1}{2}\theta r_{sensor}^2$$

Setting the distance $\triangle d$ between the "looks" to the radius of the sensor, r_{sensor}, simplifies the equation to

$$A_{wedge} = r_{sensor}^2\cos^{-1}\left(\frac{1}{2}\right). \tag{D.2}$$

The area of the triangle region YZS_1 is calculated by

$$A_{\triangle YZS_1} = \frac{d}{2}\sqrt{r_{sensor}^2 - \frac{1}{4}d^2}$$

with the substitution of $d = r_{sensor}$ yielding

$$A_{\triangle YZS_1} = \frac{\sqrt{3}r_{sensor}^2}{4} \tag{D.3}$$

Substitution and simplification of (D.1) reveals the area of double coverage between two subsequent waypoints of spacing r.

$$A_{lens} = \left(2\cos^{-1}\left(\frac{1}{2}\right) - \frac{\sqrt{3}}{2}\right)r_{sensor}^2$$

or

$$A_{lens} = 1.2284r_{sensor}^2 \tag{D.4}$$

Finally, the total area covered ($A_{covered}$) during a single leg of the discretized search can be calculated and compared to the total area that would be covered by the perfect continuous search on that same leg (A_{search}) to determine the percentage of coverage (P_d) achieved through

discretization of the searcher movement.

$$A_{covered} = \frac{l_j}{r_{sensor}} \left(\pi r_{sensor}^2 - 1.2284 r_{sensor}^2 \right)$$

$$A_{search} = 2 l_j r_{sensor}$$

$$P_d = \frac{A_{covered}}{A_{search}} = \frac{\frac{l_j}{r_{sensor}} \left(\pi r_{sensor}^2 - 1.2284 r_{sensor}^2 \right)}{2 l_j r_{sensor}}$$

$$P_d = 0.9566 \tag{D.5}$$

Given the benefits of simulation run time and data storage utilizing this discrete movement method, and the relative size of the search area compared to the area covered on each search leg, a 96% coverage ratio is sufficient for use in this simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] The Boeing Company, "Boeing Laser Systems Destroy Unmanned Aerial Vehicles In Tests," November 2009, [Press release]. [Online]. Available: http://boeing.mediaroom.com/index.php?s=43&item=941

[2] Y. Pei and B. Song, "Method for assessing unmanned aerial vehicle vulnerability to high-energy laser weapon," *Journal of Aircraft*, vol. 49, no. 1, pp. 319–323, 2012. [Online]. Available: http://doi.aiaa.org/10.2514/1.C031376

[3] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proceedings of 1993 International Conference on Advanced Robotics*, 1993, pp. 533–538.

[4] H. Choset, "Coverage for robotics–A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001. [Online]. Available: http://link.springer.com/article/10.1023%2FA%3A1016639210559?LI=true#

[5] N. Mansour, Ed., *Search Algorithms and Applications*. Rijeka, Croatia: InTech, 2011.

[6] L. Lovász, "Random walks on graphs: A survey," Yale University Department of Computer Science, Tech. Rep., 1993. [Online]. Available: http://www.cs.yale.edu/publications/techreports/tr1029.pdf

[7] D. Aldous, "An introduction to covering problems for random walks on graphs," *Journal of Theoretical Probability*, vol. 2, no. 1, pp. 87–89, 1989. [Online]. Available: http://www.springerlink.com/index/10.1007/BF01048271

[8] J. Kahn, N. Linial, N. Nisan, and M. Saks, "On the cover time of random walks on graphs," *Journal of Theoretical Probability*, vol. 2, no. 1, pp. 121–128, 1989. [Online]. Available: http://www.springerlink.com/index/LM67442534G7T5G5.pdf

[9] L. D. Stone, "Search theory: A mathematical theory for finding lost objects," *Mathematics Magazine*, vol. 50, no. 5, pp. 248–256, 1977. [Online]. Available: http://www.jstor.org/stable/2689530

[10] A. Dembo, Y. Peres, J. Rosen, and O. Zeitouni, "Cover times for Brownian motion and random walks in two dimensions," *Annals of Mathematics*, vol. 160, pp. 433–464, 2004. [Online]. Available: http://www.jstor.org/stable/10.2307/3597219

[11] M. Vahabi, J. Schulz, B. Shokri, and R. Metzler, "Area coverage of radial Levy flights with periodic boundary conditions," Cornell University, Tech. Rep., 2012. [Online]. Available: http://arxiv.org/abs/1211.1849

[12] H. Koyama, H. Sato, and A. Namatame, "Relation between waiting time and flight length for efficient search," in *Proceedings of International Conference on Instrumentation, Control and Information Technology (SICE 2008)*, no. 2, Cofu City, Tokyo, Japan, 2008, pp. 428–432.

[13] E. Gelenbe, N. Schmajuk, J. Staddon, and J. Reif, "Autonomous search by robots and animals: A survey," *Robotics and Autonomous Systems*, vol. 22, pp. 23–34, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889097000146

[14] A. M. Edwards, "Using likelihood to test for Levy flight search patterns and for general power-law distributions in nature." *The Journal of Animal Ecology*, vol. 77, no. 6, pp. 1212–22, 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18631370

[15] F. Bartumeus, J. Catalan, U. Fulco, M. Lyra, and G. Viswanathan, "Optimizing the encounter rate in biological interactions: Levy versus Brownian strategies," *Physical Review Letters*, vol. 88, no. 9, pp. 097 901–1 to 097 901–4, 2002. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11864054

[16] Y. Hu, J. Zhang, D. Huan, and Z. Di, "Toward a general understanding of the scaling laws in human and animal mobility," *EPL (Europhysics Letters)*, vol. 96, no. 3, pp. 1–8, 2011. [Online]. Available: http://iopscience.iop.org/0295-5075/96/3/38006

[17] M. Buchanan, "Ecological modelling: The mathematical mirror to animal nature," *Nature*, vol. 453, no. June, pp. 714–716, 2008. [Online]. Available: http://www.nature.com/news/2008/080604/full/453714a.html?s=news_rss

[18] G. Viswanathan, V. Afanasyev, S. Buldyrev, S. Havlin, M. da Luz, E. Raposo, and H. Stanley, "Levy flights in random searches," *Physica A: Statistical Mechanics*

and its Applications, vol. 282, pp. 1–12, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437100000716

[19] G. Viswanathan, S. Buldyrev, V. Afanasyev, S. Havlin, M. da Luz, E. Raposo, and H. Stanley, "Levy flights search patterns of biological organisms," *Physica A: Statistical Mechanics and its Applications*, vol. 295, pp. 85–88, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437101000577

[20] G. Viswanathan, V. Afanasyev, S. Buldyrev, E. Murphy, P. Prince, and H. Stanley, "Levy flight search patterns of wandering albatrosses," *Nature*, vol. 381, pp. 413–415, 1996. [Online]. Available: http://www.nature.com/nature/journal/v381/n6581/abs/381413a0.html

[21] G. Viswanathan, S. Buldyrev, S. Havlin, M. da Luz, E. Raposo, and H. Stanley, "Optimizing the success of random searches," *Nature*, vol. 401, no. 6756, pp. 911–914, 1999. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/10553906

[22] E. Raposo, S. Buldyrev, M. da Luz, M. Santos, H. Stanley, and G. Viswanathan, "Dynamical robustness of Levy search strategies," *Physical Review Letters*, vol. 91, no. 24, p. 240601, 2003. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.91.240601

[23] D. Calitoiu, "New search algorithm for randomly located objects : A non-cooperative agent based approach," in *Proceedings of 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)*, 2009, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5356564&tag=1

[24] A. Flenner, J. Flenner, J. Bobinchak, D. Mercier, A. Le, K. Estabridis, and G. Hewer, "Levy walks for autonomous search," in *Proceedings of SPIE Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III*, vol. 8389, 2012, p. 83890. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.918719

[25] E. Pantaleo, P. Facchi, and S. Pascazio, "Simulations of Levy flights," *Physica Scripta*, vol. 2009, p. 014036, 2009. [Online]. Available: http://stacks.iop.org/1402-4896/2009/i=T135/a=014036?key=crossref.4c5cde6c31b29ffdafe7bfa2aabf4786

[26] W. Lenagh and P. Dasgupta, "Levy distributed search behaviors for mobile target locating and tracking," in *Proceedings of 19th Conference on Behavior Representation in Modeling and Simulation*, 2010, pp. 103–109.

[27] R. Mantegna and H. Stanley, "Stochastic process with ultraslow convergence to a Gaussian: The truncated Levy flight," *Physical Review Letters*, vol. 73, no. 22, pp. 2946–2949, 1994. [Online]. Available: http://cps-www.bu.edu/hes/articles/ms94.pdf

[28] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957. [Online]. Available: http://www.jstor.org/stable/10.2307/2372560

[29] A. Furtuna, D. J. Balkcom, H. Chitsaz, and P. Kavathekar, "Generalizing the Dubins and Reeds-Shepp cars: Fastest paths for bounded-velocity mobile robots," in *Proceedings of 2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 2533–2539. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543594

[30] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990. [Online]. Available: http://stat.wharton.upenn.edu/~shepp/publications/99.pdf

[31] D. Anisi, J. Hamberg, and X. Hu, "Nearly time-optimal paths for a ground vehicle," *Journal of Control Theory and Applications*, vol. 1, pp. 2–8, 2003. [Online]. Available: http://www.springerlink.com/index/xk45698hx747572j.pdf

[32] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *Proceedings of 46th IEEE Conference on Decision and Control*, 2007, pp. 2379–2384. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4434966

[33] G. Parlangeli, L. Ostuni, L. Mancarella, and G. Indiveri, "A motion planning algorithm for smooth paths of bounded curvature and curvature derivative," in *Proceedings of 17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece, 2009, pp. 73–78. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5164517

[34] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, pp. 179–202, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889000001275

[35] T. H. Chung, M. Kress, and J. O. Royset, "Probabilistic search optimization and mission assignment for heterogeneous autonomous agents," in *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 939–945. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5152215

[36] P. J. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995. [Online]. Available: http://www.stat.duke.edu/~scs/Courses/Stat376/Papers/TransdimMCMC/GreenRevJump.1995.pdf

[37] M. Kress and R. Szechtman, "Efficient employment of non-reactive sensors," *Military Operations Research*, vol. 13, no. 4, pp. 45–57, 2008. [Online]. Available: http://www.ingentaconnect.com/content/mors/mor/2008/00000013/00000004/art00006

[38] T. H. Chung and J. W. Burdick, "A decision-making framework for control strategies in probabilistic search," in *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 4386–4393. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4209773

[39] T. Stevens and T. Chung, "Autonomous search and counter-detection using Levy search models," in *Proceedings of 2013 International Conference on Robotics and Automation*, 2013, (to appear).

[40] A. R. Washburn, *Search and Detection, 4th Edition*. Institute for Operations Research and the Management Sciences, 2002.

[41] "Continuous Detection Models," class notes for OA3602: Search Theory and Detection, Department of Operations Research, Naval Postgraduate School, Winter, 2012.

[42] "Random Coverage Models," class notes for OA3602: Search Theory and Detection, Department of Operations Research, Naval Postgraduate School, Winter, 2012.

[43] Headquarters, Department of the Army, "U.S. ARMY FMI 3-04.155 Army Unmanned Systems Operations," United States Army, Tech. Rep., 2006. [Online]. Available: https://www.fas.org/irp/doddir/army/fmi3-04-155.pdf

[44] R. Wright, G. Klager, and F. Herbst, "Application of small unmanned air vehicles in network centric warfare," U.S. Army Research, Development and Engineering Command, Tech. Rep., 2005. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA471933

[45] Wolfram Research, "Wolfram Mathematica 9 Documentation Center," January 2013, Accessed: 2/15/2013. [Online]. Available: http://reference.wolfram.com/mathematica/ref/LevyDistribution.html

[46] J. Devore, *Probability & Statistics for Engineering and the Sciences*. Boston, MA: Brooks/Cole, 2012.

[47] "Consortium for Robotics and Unmanned Systems Education and Research," Accessed: 03/01/2013. [Online]. Available: http://cruser.nps.edu

[48] Lockheed Martin, "Lockheed Martin Procerus Technologies," Accessed: 03/01/2013. [Online]. Available: http://procerusuav.com

[49] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, pp. 299–316, Nov. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10514-011-9241-4

[50] D. Calitoiu and D. Milici, "Modeling with non-cooperative agents: Destructive and non-destructive search algorithms for randomly located objects," in *Proceedings of 2010 Summer Computing Simulation Conference*, 2010, pp. 102–109. [Online]. Available: http://dl.acm.org/citation.cfm?id=1999427

[51] N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. Tuttle, "Many random walks are faster than one," in *Proceedings of the 20th Annual Symposium on Parallelism in Algorithms and Architectures*, 2008, pp. 1–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1378557

[52] R. Spence and S. Hutchinson, "An integrated architecture for robot motion planning and control in the presence of obstacles with unknown trajectories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 1, pp. 100–110, 1995. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=362962

[53] F. Rafi, S. Khan, K. Shafiq, and M. Shah, "Autonomous target following by unmanned aerial vehicles," in *Proceedings of SPIE Unmanned Systems Technology VIII*, vol.

6230, May 2006, p. 623010. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1286517

[54] S. Ponda, R. Kolacinski, and E. Frazzoli, "Trajectory optimization for target localization using small unmanned aerial vehicles," Massachusetts Institute of Technology, Tech. Rep., 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.208.360

[55] M. gyu Park, J. hyun Jeon, and M. cheol Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *Proceedings of IEEE International Symposium on Industrial Electronics*, vol. 3, 2001, pp. 1530–1535. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=931933

[56] S. Waharte, A. Symington, and N. Trigoni, "Probabilistic search with agile UAVs," in *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2840–2845. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5509962

[57] K. Savla, F. Bullo, and E. Frazzoli, "The coverage problem for loitering Dubins vehicles," in *Proceedings of 2007 46th IEEE Conference on Decision and Control*, 2007, pp. 1398–1403. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4435017

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia
6. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
7. CAPT Jeff Kline, USN(r)
   Naval Postgraduate School
   Monterey, California
8. Dr. Jim Law
   SPAWAR SSC-Pacific
   San Diego, California